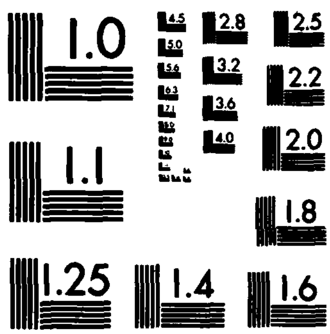
AD-A127 398 SOFTWARE DESCRIPTION FOR THE O'HARE RUNWAY 1/3
CONFIGURATION MANAGEMENT SYSTE..(U) MITRE CORP MCLEAN
VA METREK DIV S KAVOUSSI OCT 82 MTR-82W125-VOL-1
UNCLASSIFIED FAA-EM-82-28-VOL-1 DTFA01-81-C-10003 F/G 1/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Report No. FAA-EM-82-28
Vol. I

MTR-82W125
Vol. I

SOFTWARE DESCRIPTION FOR THE O'HARE RUNWAY CONFIGURATION MANAGEMENT SYSTEM

VOLUME I: TECHNICAL DESCRIPTION

SADEGH KAVOUSSI

The MITRE Corporation
McLean, Virginia 22102

OCTOBER 1982

Document is available to the U.S. public through
the National Technical Information Service
Springfield, Virginia 22161

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
OFFICE OF SYSTEMS ENGINEERING MANAGEMENT
Washington, D.C. 20591

83

N O T I C E

Technical Report Documentation Page

1. Report No. FAA-EM-82-28 Volume I	2. Government Accession No. A127 398	3. Recipient's Catalog No.
4. Title and Subtitle Software Description for the O'Hare Runway Configuration Management System Volume I: Technical Description		5. Report Date October 1982
		6. Performing Organization Code
7. Author(s) Sadegh Kavoussi		8. Performing Organization Report No. MTR-82W125 Volume I
9. Performing Organization Name and Address The MITRE Corporation Metrek Division 1820 Dolley Madison Boulevard McLean, Virginia 22102		10. Work Unit No. (TRAIS)
		11. Contract or Grant No. DTFA01-81-C-10003
12. Sponsoring Agency Name and Address Department of Transportation Federal Aviation Administration Office of Systems Engineering Management Washington, D. C. 20591		13. Type of Report and Period Covered
		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

This document describes the software developed as part of the Chicago O'Hare Runway Configuration Management System (CMS). The software is designed as an interactive automated planning aid to assist the O'Hare assistant chief in the consistent selection of efficient runway configurations in order to lower aircraft delays. In addition, CMS serves an an information management system by consolidating various airport data and making them available for the O'Hare facility personnel. Volume I of this document contains the general description of the CMS software plus high level pseudocodes describing its logic. Volume II is dedicated to detailed description of the software via low level pseudocodes.

17. Key Words	18. Distribution Statement Document is available to the public through the National Technical Information Service, Springfield, VA 22161		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

ACKNOWLEDGEMENT

The author is indebted to Lucille Perrin for the many hours spent in preparing this document.

EXECUTIVE SUMMARY

The O'Hare Runway Configuration Management System (CMS) is an interactive multi-user computer system designed to aid O'Hare management personnel in the consistent selection of runway configurations in order to reduce aircraft delays. CMS is also used for the purpose of communicating and disseminating information about the airport among the tower and Terminal Radar Control Facility (TRACON) personnel.

Although the CMS software was written for O'Hare International Airport, it can be adapted for other airports to serve as an automated planning aid for runway configuration management. This would require changing the associated site specific adaptation data. At some airports, however, the need might be to manage the surrounding airspace which is shared with other airports, or to manage the flow of aircraft on taxiways as opposed to runway configuration management. The basic concepts of CMS can be extended to include such applications as well but would require site specific model development to suit the needs of the individual airport.

The purpose of this report is to describe the CMS software in the time sharing environment of MITRE Washington's Computer Center. Currently, CMS is housed in an IBM 4341 computer with VM/SP operating system. CMS employs the IBM's Display Management System (DMS) software package that provides full screen menu type displays. The display terminals used by CMS are IBM's 3270 series or equivalent. The CMS software is written exclusively in PL/I and complies fully with top-down structured programming techniques.

CMS has been designed to facilitate manual data entry, since automated inputs are not yet readily available. CMS is available for interactive access by the tower and radar room personnel who normally monitor and report changes in the airport operational environment. These users are: the Assistant Chief (AC), who has the primary responsibility for configuration selection; the team supervisor of the tower cab (CAB), who provides operational information (wind, weather, runway conditions) to the system; and the Airways Facilities operations officer (AF), who is responsible for the runway equipment status. The interactions between these users and CMS are illustrated in Figure A.

Because of the limitations of the time-sharing system under which CMS is currently operating, these three different users can only be supported by three separate programs. These programs are compiled and stored separately and operate independently, but communicate through a common data base which contains all information on O'Hare status over the planning horizon. When CMS is implemented at

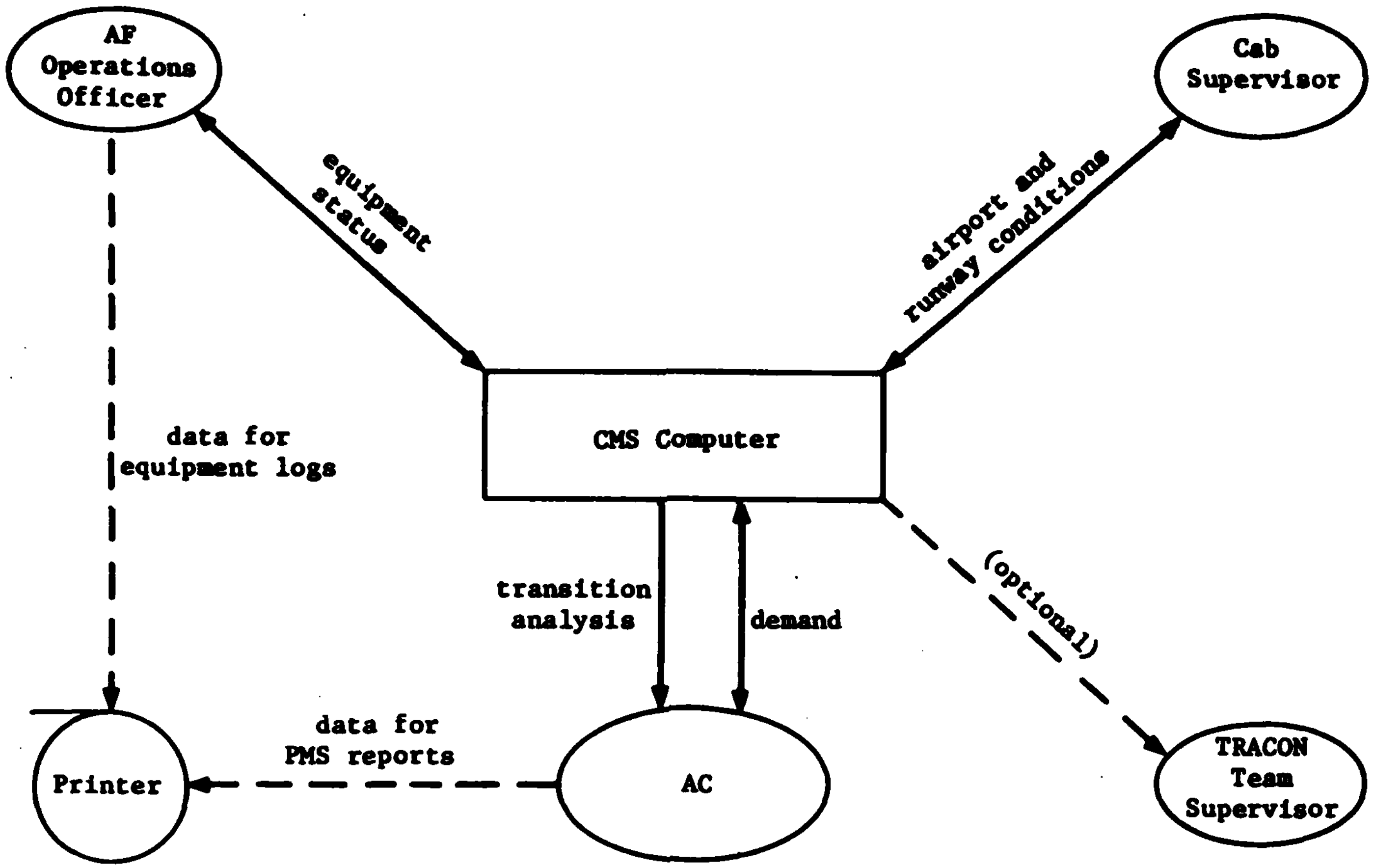


FIGURE A
PHYSICAL CONFIGURATION OF CMS

O'Hare, it will operate on a dedicated mini-computer which permits multi-tasking (that is, multiple users interacting with a single program simultaneously). This will eliminate the need for three separate programs; certain changes to the program structure will be required to make best use of the multi-tasking environment, but the basic CMS logic will not be affected.

Each program within the CMS software package supports a set of data "screens", each containing a predetermined subset of information for input or display. An example of a CMS screen is given in Figure B. Table A contains a list of display screens within the CMS software. In some cases there is an overlap of information among several screens. Although the screens are not mutually independent (i.e., changes in one screen may affect the contents of the others), they are self-contained in that they serve a specific purpose and are acted upon separately.

The screens provide a convenient format for entering data on the current and future operating environment at O'Hare. This includes information on wind speed and direction, ceiling and visibility, runway surface conditions, status of runway landing aids, and the expected volume and distribution of traffic. This information is then used by CMS to determine the operational availability of individual runways. The operational suitability of the runway configurations is then determined, based upon runway availability and other operational factors; and the configurations are ranked according to their capacities, based upon projected demand for the next hour. The penalty of transitioning from current configuration, in terms of capacity during the transition period, is also calculated and displayed. This yields the primary output of the runway configurations management system -- an ordered list of transition strategies indicating which runways to use at what times during the planning period.

Volume I of this report defines the major subsystems within the CMS software package, discusses the overall control and architecture of the CMS software, and describes the software logic pertaining to each component. "High-level", English-like pseudocode is used to describe the CMS software. Pseudocode is used because it can provide a clear, English-like description which is believed superior to flowcharts for conveying complex logic to the reader, while still maintaining a formal structure.

Volume II contains the "low-level", variable specification pseudocode, in order to provide a detailed description of the software.

RUNWAY EQUIPMENT PLANNING LOG

RWY	EQUIPMENT	OTS	RTS	REMARKS
4L	ALS	1500	1600	REPAIRS

ASST. CHIEF---ADDITIONAL ENTRIES

DATA STORED AT 1447

FIGURE B
EXAMPLE OF CMS SCREEN

TABLE A
LIST OF INPUT/OUTPUT DISPLAY SCREENS

1.	Menu of Program Function Keys and Program Termination
2.	Parameters
3.	O'Hare Status Summary
4.	Planning Log Selection
5.	Wind and Weather Planning Log
6.	Runway Conditions Planning Log
7.	Equipment Planning Log
8.	Demand Planning Log
9-10.	Airport Status (Current/Forecast)
11-12.	Runway Equipment Status (Current/Forecast)
13-14.	Demand Profile (Current/Forecast)
15-16.	Ordered List of Configurations (Current/Forecast)
17.	Current Departure Queues
18.	Ordered List of Transitions
19-20.	Configuration Information (Current/Forecast)

TABLE OF CONTENTS

TABLE OF CONTENTS
(Concluded)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF ILLUSTRATIONS
(Concluded)

1. INTRODUCTION

1.1 Purpose and Scope

The O'Hare Runway Configuration Management System (CMS) is an interactive multi-user computer system designed to aid the O'Hare management personnel in the consistent selection of runway configurations in order to reduce aircraft delays. CMS is also used for the purpose of communicating and disseminating information about the airport among the various tower and Terminal Radar Control Facility (TRACON) personnel (Reference 1).

The purpose of this report is to describe the CMS software as it stands in the time sharing environment of MITRE Washington's Computer Center. In these documents (Volumes I and II), the major subsystems within the CMS software package are defined and the software logic pertaining to each component is described fully. Also, the overall control and architecture of the CMS software is discussed.

Additional information on the use of CMS and logic details may be found in References 2 and 3.

1.2 Use of Pseudocodes

In these documents, pseudocodes are used to describe the CMS software in detail. The pseudocode approach to software documentation is adopted because it provides a clear, English-like description, which is believed superior to flowcharts for conveying complex logic to the reader, while still maintaining a formal structure. In order to present different levels of detail to all readers, two types of pseudocodes are used: "high-level" English-like and "low-level" variable specification.

There exist a number of different pseudocode languages, each suited for a particular application or style. For this document the pseudocode language "E" or "Eclectic" was chosen. "E" was developed by The MITRE Corporation in conjunction with the Air Traffic Advisory and Resolution Service (ATARS) project (References 4 and 5). The decision to choose "E" was made based on the fact that "E" is designed to support PL/I programming language, is suited for structured programming style, and has readily accessible documentation. Appendix B presents a brief summary of the syntax of "E" language.

1.3 Organization

In Volume I, a brief overview of CMS is presented in Section 2. Section 3 discusses a number of software design considerations and limitations associated with the current version of CMS. The overall structure of the CMS software is described in Section 4. Section 5 familiarizes the reader with the CMS data base. Sections 6 and 7 present the logic of CMS software; the former section deals with system data structures and top level processing while the latter describes individual screens. The logic of the CMS software is presented in the form of high level pseudocodes in Appendix A. Finally, there are a number of other Appendices that deal with topics such as syntax of E language, CMS utility programs, PL/I built-in functions used in CMS, and data base format.

In Volume II, a more detailed description of the CMS software is presented in the form of low-level pseudocodes.

2. CMS OVERVIEW

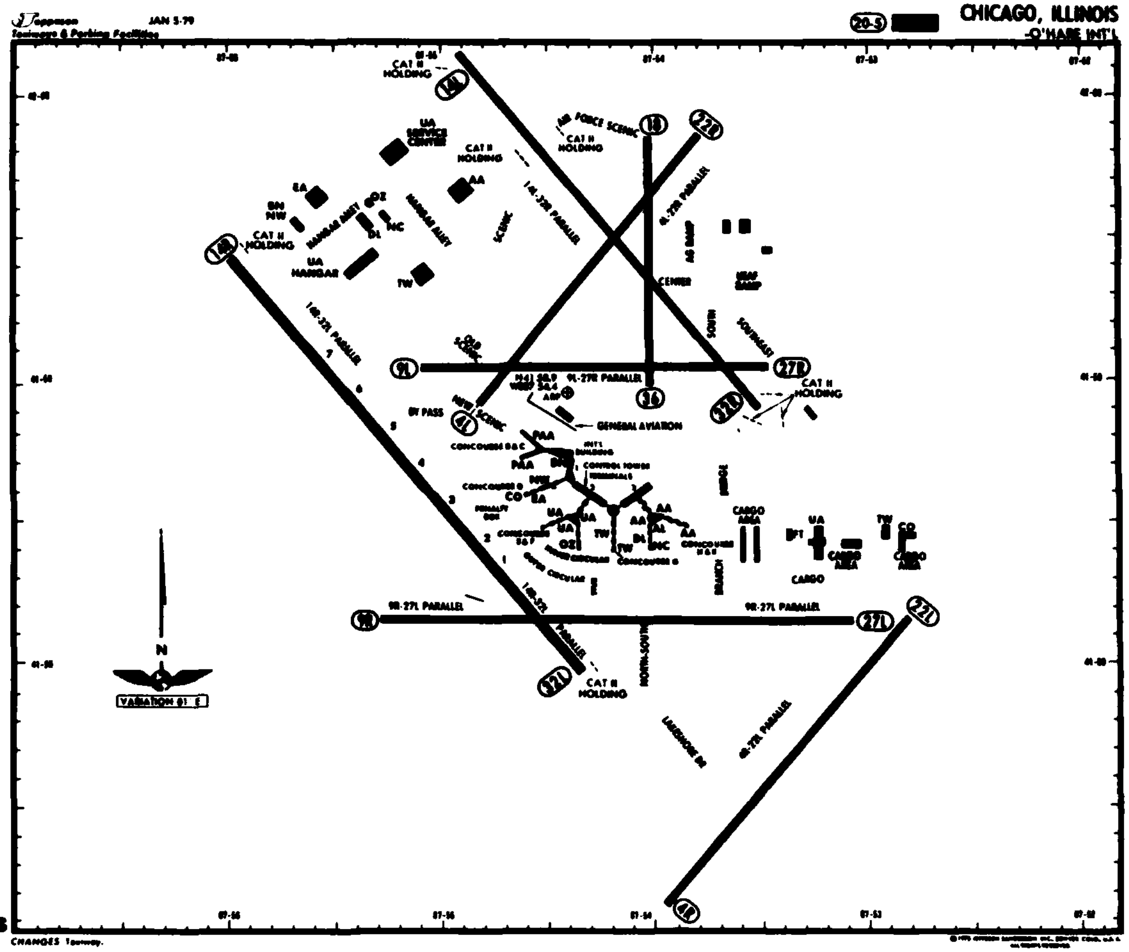
This section offers an overview of the O'Hare Configuration Management System. The description has been adapted and summarized from Reference 1.

The O'Hare Runway Configuration Management System (CMS) is an interactive computer program designed to aid the combined O'Hare tower/TRACON facility management in the consistent selection of runway configurations in order to lower aircraft delays.

At O'Hare, the process of runway configuration selection is complex because of the runway layout (Figure 2-1) and the dynamic nature of airport operations. There are twelve main runway ends and a short runway (18/36) which is used solely for general aviation traffic under visual conditions. Using only twelve main runways, O'Hare personnel have identified seventy-three operationally feasible runway configurations that use at least two arrival/departure runway pairs simultaneously. Additionally, there are a great number of runway combinations that include fewer runways. Moreover, O'Hare is one of the major connectors of domestic and international flights causing significant variations in the volume and distribution of air traffic over each of its arrival and departure fixes. Furthermore, rapidly changing weather and wind conditions in the Chicago area further complicate the process of runway configuration selection. The above mentioned problems, plus others common to all major airports such as runway closures and equipment outages make CMS a useful planning aid for O'Hare.

Today at O'Hare, the Assistant Chief (AC) has the primary responsibility for configuration selection. The decision on changing runway configurations is based primarily on airport status, equipment status, and traffic demand, and requires extensive coordination with supervisors of the tower cab and the TRACON. CMS provides the means to consolidate and display information relevant to this decision making process. Furthermore, CMS automatically integrates information on airport status, equipment status, and traffic demand into a measure of capacity for evaluating different configuration choices, as well as provides the AC with a tool in planning transitions between the currently operating configuration and the set of feasible ones under forecast conditions.

The elements that make up the runway configuration management process are depicted in Figure 2-2. The initial step in runway configuration management is to define the respective current and forecast scenarios. In the case of the current scenario, this



BASE DRAWING PROVIDED BY
JEPPESEN & CO.
ILLUSTRATION ONLY—NOT TO BE
USED FOR NAVIGATIONAL PURPOSES

FIGURE 2-1
CHICAGO O'HARE INTERNATIONAL AIRPORT

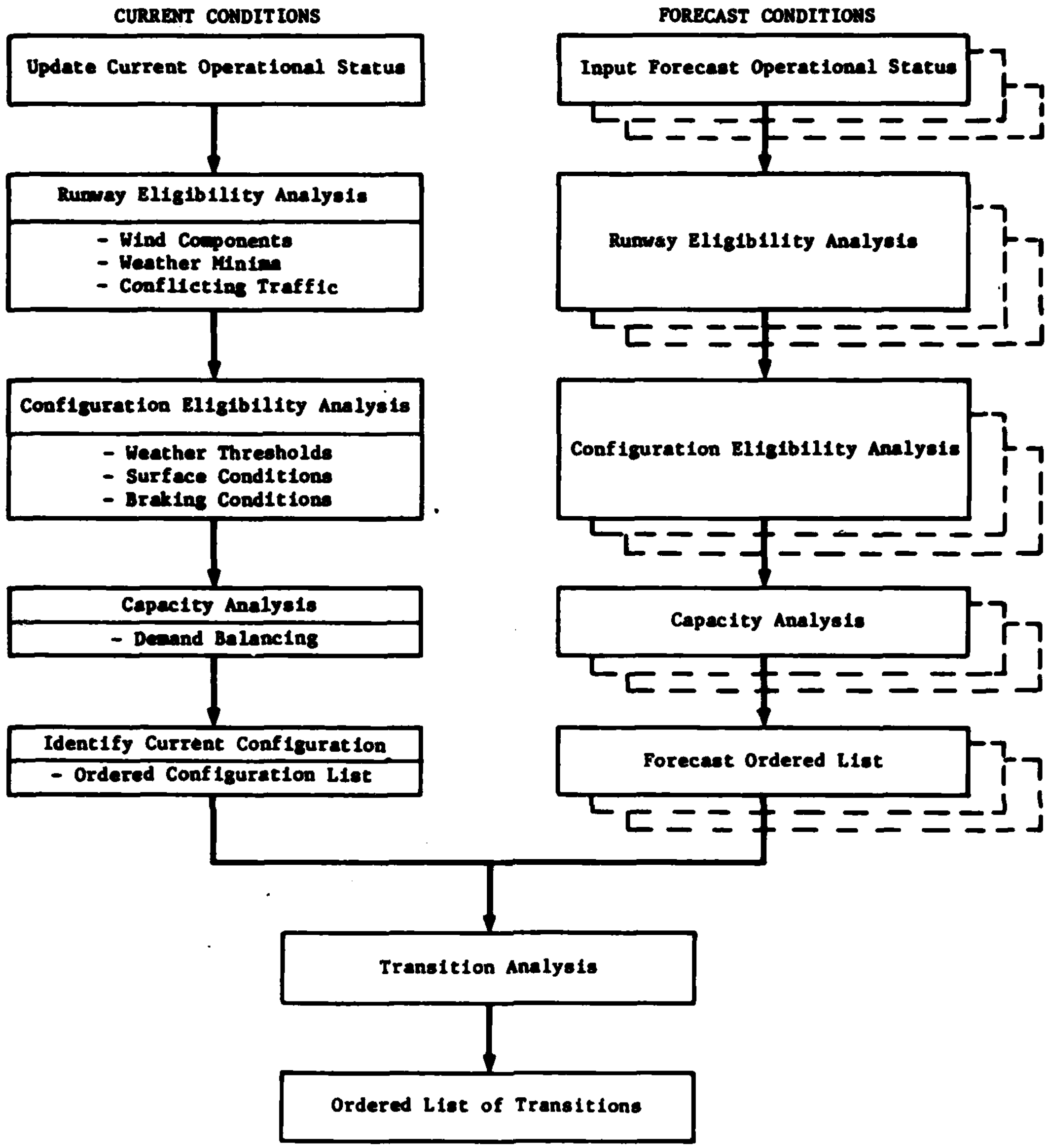


FIGURE 2-2
MAJOR ELEMENTS OF RUNWAY CONFIGURATION MANAGEMENT

is accomplished by making sure that CMS is continually updated with all changes in the current operating environment. In the absence of automated interfaces with monitoring systems in the tower and TRACON facilities, the updating functions of current operating conditions can be delegated to those who now have the responsibility for monitoring and reporting that information without creating additional workloads. Such responsibilities today lie with the cab team supervisor who is in charge of updating weather, wind, and runway conditions and Airway Facilities (AF) operations officer who maintains the runway equipment status.

On the other hand, the forecast scenarios reflect the changes in the operational environment expected to occur in the future that may be substantial enough to require configuration changes. The responsibility in this case lies with the AC who acquires the forecast conditions and expected changes through a system of interconnected planning logs communicated to him by other participants (AF operations officer and cab team supervisors). Therefore, the AC constructs forecast scenarios from actual forecast conditions or other hypothesized situations as is deemed necessary.

The next step in runway configuration management is to determine the operational availability of individual runways and subsequently configurations within each scenario (current and forecast). This is accomplished by runway and configuration eligibility analyses performed by CMS. After a list of eligible configurations is determined, CMS performs capacity analysis and produces ordered lists of configurations based on capacity within each scenario.

Finally, the last step in runway configuration management is the transition analysis. CMS combines the current and forecast scenarios to produce an ordered list of transitions. The transition analysis helps the AC in selecting configurations that have lower transition penalties and high capacities.

A proposed CMS hardware configuration consists of a central computer supporting three CRT display terminals and one printer (Figure 2-3). A terminal is dedicated to the AF operations officer and another to the supervisor of the tower cab, while the third terminal is used by the AC. Each terminal allows a selected set of inputs into CMS consistent with the information for which that terminal position is responsible as will be defined later in this document.

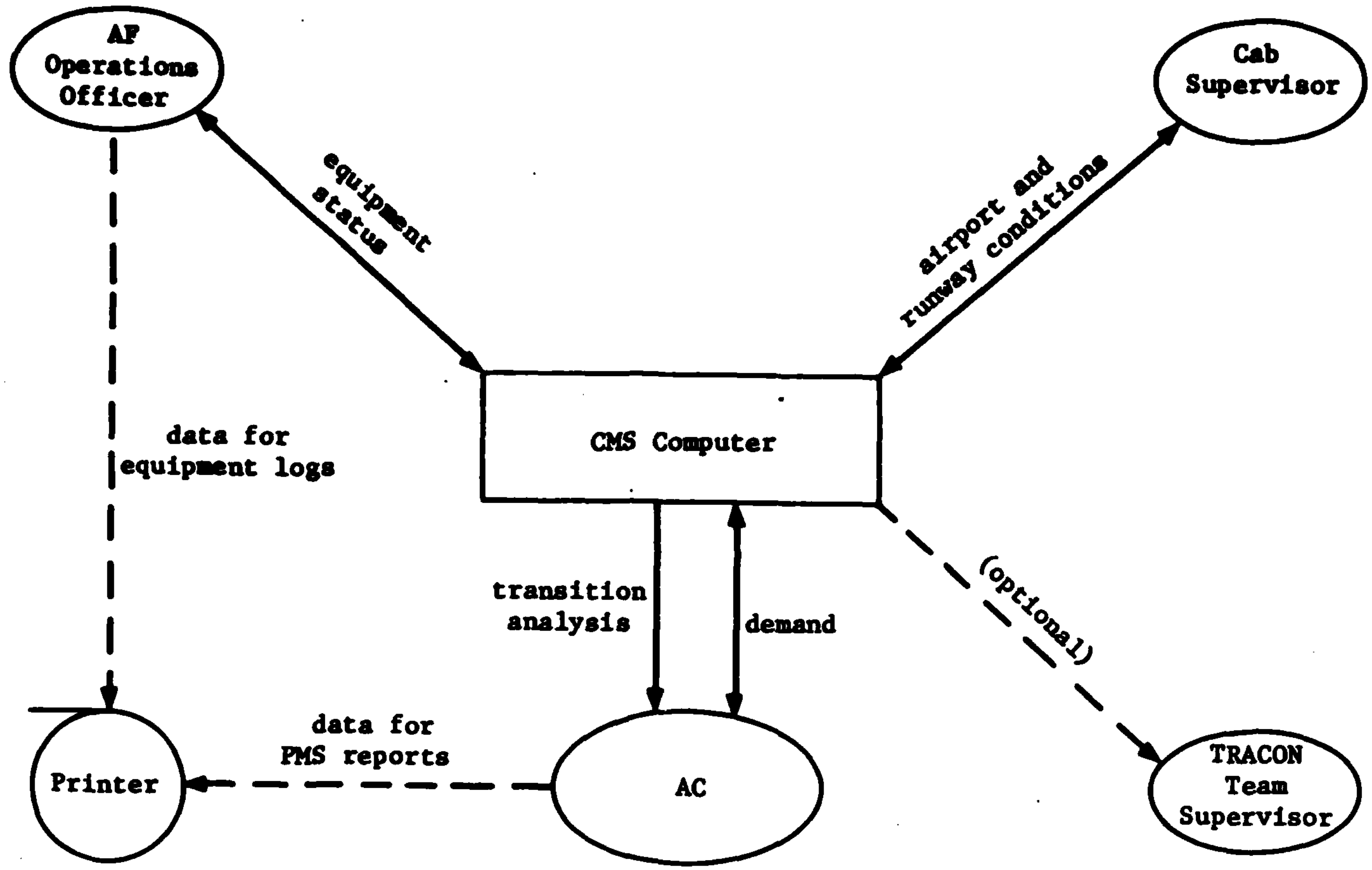


FIGURE 2-3
PHYSICAL CONFIGURATION OF CMS

CMS offers a number of other benefits in addition to its primary purpose of assisting the AC in runway configuration selection. The use of multiple terminals to access a common central data base provides the facility personnel with immediate displays of the current operational status of the airport. The multiple terminal concept also allows direct communication between different users of CMS. This reduces workloads related to telephone communication and paperwork. Also, the hard copy option provided by CMS could be useful in the generation of logs and historical records, automating such work currently being performed manually by O'Hare personnel.

3. SOFTWARE DESIGN CONSIDERATIONS

This section deals with the discussion of several design factors considered for the CMS software in its current form residing in the time sharing environment of MITRE Washington's Computer Center. Currently, CMS is housed in an IBM 4341 computer with VM/SP operating system. CMS employs the IBM's Display Management System (DMS) software package that provides full screen menu type displays. The display terminals used by CMS are IBM's 3270 series or equivalent.

3.1 General Design Requirements

In order to interface with DMS software package one of the following programming languages was needed: COBOL, RPGII, PL/I, or Assembler. The DMS language requirement, as well as the need for a high level programming language suited for scientific applications have led to the choice of PL/I as the programming language for CMS. Hence, the CMS software is written exclusively in PL/I and complies fully with top-down structured programming techniques.

In its final implementation at O'Hare, CMS is envisioned to have direct interfaces with existing and future monitoring systems providing automated inputs that would greatly reduce the user participation. However, in the absence of such automated interfaces, the current version of CMS requires manual inputs by the participants. As a result, care has been taken to make the system as user friendly as possible without increasing the workloads of its users. In working towards these goals, the multi-terminal (user) concept has been introduced in order to increase the efficiency of input process by spreading it among several users.

Besides the reduction of workload resulting from spreading of the input process among several users, CMS offers other features that make it a user friendly system. These features are mostly in the area of input/output interfaces to CMS enhancing the ease with which the users interact with the system.

One of these areas is the invocation process of various CMS functions. For the purpose of invoking the various CMS functions, the excessive use of keyboard strokes and/or the need for command oriented conversational languages have been eliminated. Instead, in order to initiate each CMS function, a separate key on the terminal keyboard is used requiring only a single stroke. These keys are referred to here and throughout this document as program function (PF) keys. In order to remind the users of the definition of various PF keys, CMS displays a list of all the PF keys and their associated functions when it is activated, and thereafter upon request. However, in the final implementation of CMS the program function keys on the terminal keyboard will be labeled with the name or description of the function they represent.

Another feature offered by CMS is in the area of the man-machine interface. In order to assist the users in input/output processes, CMS provides menu type input/output screens (Figure 3-1) . These screens are designed and formatted so as to eliminate extraneous effort on the part of the users by displaying the description of the required inputs and outputs. In case of inputs, CMS screens provide space to move the cursor onto and type at most a single number or a symbol to initiate a particular input process. Menu type screens are particularly useful since they are self-explanatory, easy to learn, and simple to operate. Additionally, CMS menu type screens provide the ability to display selected information in high intensity for more emphasis.

In addition to the above features, CMS contains error checking and data validation routines for each screen that prevent the user from accidently entering erroneous data. If the user makes an error on the screen, the input is not accepted by CMS, and a highlighted message instructs him with corrective measures and places the cursor on the faulty entry. This capability of CMS is helpful in not only preventing erroneous and invalid data entries, but in serving as a self-training process for the user. As the user encounters various error messages, he becomes more familiar with the system and learns how to use it more effectively.

3.2 Design Limitations of a Time-Share System

There are a number of limitations associated with the design of CMS in its current form in the time sharing environment. These limitations will not exist if CMS were to be implemented on a stand-alone computer as is envisioned for O'Hare in the future.

The time sharing environment where CMS resides does not allow multi-tasking, and therefore control and communciation between several terminals via a single program is not possible. Therefore, in order to implement the multi-terminal concept associated with CMS, three separate programs were prepared.

RUNWAY EQUIPMENT PLANNING LOG

RWY	EQUIPMENT	OTS	RTS	REMARKS
4L	ALS	1500	1600	REPAIRS

ASST. CHIEF---ADDITIONAL ENTRIES

DATA STORED AT 1447

FIGURE 3-1
EXAMPLE OF CMS SCREEN

Each program is compiled and stored separately and activated from its display terminal. The use of three programs in this fashion introduces additional constraints in terms of computer resources and processing efficiency that otherwise would not exist. The three programs within the CMS software package are loaded simultaneously into the computer memory at execution time requiring more processing resources than a single program. This added to the fact that in the time sharing environment other jobs are processed simultaneously affects the CMS response time dramatically. Additionally, the cost associated with storing, maintaining, and modifying the CMS software package is increased considerably as a result of having multiple programs as opposed to a single program.

Another limitation imposed by the time sharing environment on CMS is in the area of automatic updating capability. Currently, CMS can not provide automatic updating. Updating is done on request since each program operates independently and is not aware of the activities of the other programs.

4. SOFTWARE ARCHITECTURE

4.1 Major Subsystems

CMS is a collection of three programs that are stored and compiled separately and operate independently. The programs work in parallel with each other; each accesses a central data base containing all information on O'Hare status over the planning horizon.

The programs within the CMS software package consist of a collection of screens each containing a predetermined set of information. In some cases there is an overlap of information among several screens. For example, the current operating configuration is indicated on as many as four separate screens. Although the screens are not mutually independent (i.e., changes in one screen may affect the contents of the others), they are self-contained in that they serve a specific purpose and are acted upon separately.

Each screen within a program is considered to be a subsystem, consisting of routines that prepare the displayed data, control the screen, and perform error checking and data validation functions. Table 4-1 contains a list of all the screens available within the CMS software package.

4.2 Flow of Control

There are two states associated with each program while it operates: waiting state and execution state. When the program is in the waiting state, a particular screen is displayed. As long as the user does not request an update or a new screen, the program remains in the waiting state. However, the user can continue the use of the displayed screen by changing, deleting, or adding data as desired. The program in waiting state continues to perform the error checking and data validation functions local to the displayed screen. As long as the user does not initiate an update function or request a change of screens, the displayed data remains local to the screen, and can be restored to its original form without going through the data base.

If the update function is initiated, or if a new screen is requested, the program enters the execution state. Once in the execution state, the central data base is accessed, its contents are changed by the new data obtained from the displayed screen, and then it is released. After the release of the data base is completed, a number of pertinent calculations are performed with

TABLE 4-1
LIST OF INPUT/OUTPUT DISPLAY SCREENS

1. Menu of Program Function Keys and Program Termination

2. Parameters

3. O'Hare Status Summary

4. Planning Log Selection

5. Wind and Weather Planning Log

6. Runway Conditions Planning Log

7. Equipment Planning Log

8. Demand Planning Log

9-10. Airport Status (Current/Forecast)

11-12. Runway Equipment Status (Current/Forecast)

13-14. Demand Profile (Current/Forecast)

15-16. Ordered List of Configurations (Current/Forecast)

17. Current Departure Queues

18. Ordered List of Transitions

19-20. Configuration Information (Current/Forecast) the new updated data. These include: wind components analysis, runway minima computations, runway availability analysis, configuration eligibility analysis, capacity calculations, and demand balancing. At the end of the computations referred to above, the program returns to waiting state by displaying the new requested screen or updated previous screen.

4.3 Screen Control

CMS employs IBM's Display Management System (DMS) software package to provide the means for the user to communicate with it via menu type display screens. Each of the screens within the CMS is invoked by pressing a key on the display terminal keyboard called program function key (PF key). There are twelve PF keys numbered 1 through 12 available on IBM's 3270 display terminal keyboard or equivalent currently used by CMS. Figure 4-1 shows the layout of the display terminal's keyboard and the placement of PF keys.

The screens are predetermined and formatted in advance. They each contain permanent text fields that describe the type of information required on that screen. Also, each screen provides a number of data fields where the user can input data or CMS can display the required information. Figure 4-2 contains an example of a CMS screen with its text and data fields. However, if the data fields are used for information display only, then they can be locked out so as to not permit any entries. Moreover, DMS provides the capability to display both the text and data fields in high intensity to help emphasize or bring to attention certain types of information on the screen.

The loading and unloading of the data onto and from various screens are controlled by DMS. DMS recognizes the data in character form only. Therefore, if a numerical input is required (e.g., ceiling), CMS software contains special routines that convert the data from numerical to character form and vice versa internally before and after displaying them on the screen.

Once in the waiting state, CMS awaits user's response either in the form of request for a new screen or data entry on the screen. The new data enteries are not accepted by CMS unless the ENTER key (see Figure 4-1) is pressed and no error is detected. Pressing the ENTER key activates the screen's error checking and data validation routines. These routines check each new entry individually; if an erroneous or invalid data is detected, a message of appropriate corrective action is issued and that entry is highlighted with the cursor placed on it. In some instances a new entry may cause a number of changes on other data fields on the same screen. In these cases the local

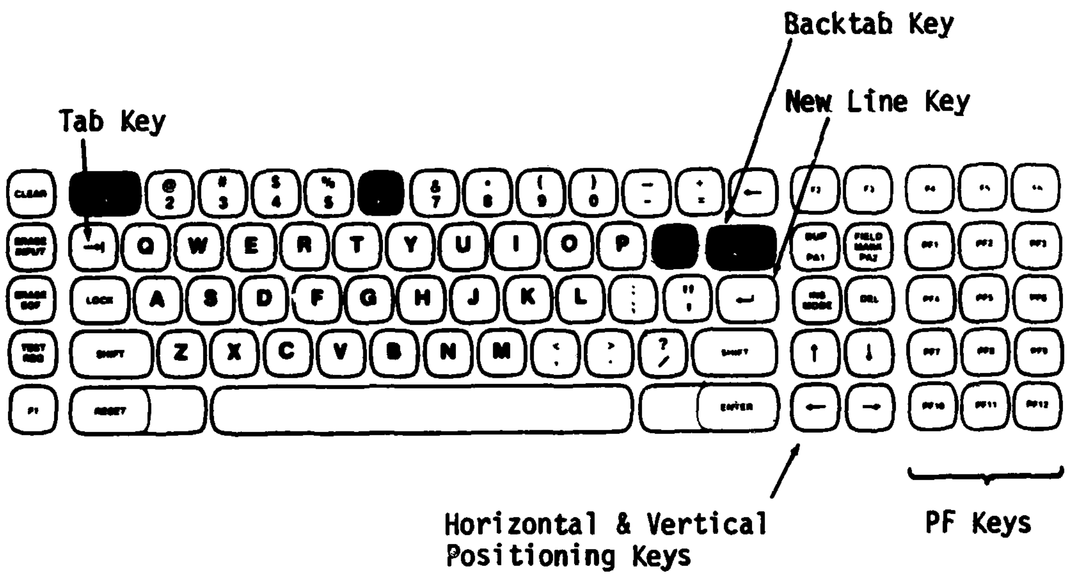


FIGURE 4-1
TYPEWRITER KEYBOARD AND
PROGRAM FUNCTION KEYS

TEXT FIELDS | DATA FIELDS

CURRENT AIRPORT STATUS

CENTERFIELD: WX: CEIL... 600 VIS... .75

WIND: DIR.... 1700 VEL... 7

MIDWAY 13R ARR IN USE...

RWY	TOWER		SURF	BRK	RVR	WIND				MINIMA		CLOSED	
	ARR	DEP	WET	POOR		DIR	VEL	CRSS	TAIL	CEIL	VIS	ARR	DEP
4R						170	7	5	5	200	.50		
4L						170	7	5	5	402	1.25	X	
9R						170	7	7	0	200	.50		
9L						170	7	7	0	200	.50		
14R						170	7	4	0	533	1.00	X	
14L						170	7	4	0	100	.25		
22R						170	7	5	0	200	2.00	X	
22L						170	7	5	0	200	.50		
27R						170	7	7	1	200	.50		
27L						170	7	7	1	200	.50		
32R						170	7	3	6	200	.50		
32L						170	7	3	6	200	.50		

DATA STORED AT 1755

FIGURE 4-2
EXAMPLE OF A SCREEN WITH ITS TEXT AND DATA FIELDS

updates on the screen occur only after the new entries have gone through the error checking and data validation routines. Figure 4-3 depicts a sample flow of control within each screen.

4.4 Differences in Structures of the Three Programs

The CMS software package is a collection of three separate programs each dedicated to a different user within the O'Hare facility. These three users are: the Assistant Chief (AC), team supervisor of the tower cab (CAB), and the Airway Facilities operations officer (AF). Since AC has the primary responsibility of configuration selection, his program provides a full access to all the available screens. In contrast, the other two programs have only a limited access to all the screens. Table 4-2 shows each program's screen availability.

Despite the differences in the number of available screens, all three programs have similar structures. However, since the CAB and AF programs do not have access to screens that perform a number of planning functions for the AC, namely ordered list of configurations and transitions screens, these programs do not contain the routines that perform the computations in the execution state (i.e., wind components, runway closure, capacity analysis, etc).

Apart from the major differences between three programs discussed above, there are some minor differences associated mostly with the individual screens accessed by more than one program. The planning log screens (i.e., wind and weather, equipment, runway conditions) contain predesignated data entry areas for each user. Although the same planning log screens are accessed by different users, each user makes entries on different part of these screens designated to him. Also, the AF and CAB have access to a number of screens that are for display only purposes, and are designed to lock out any attempted data entries.

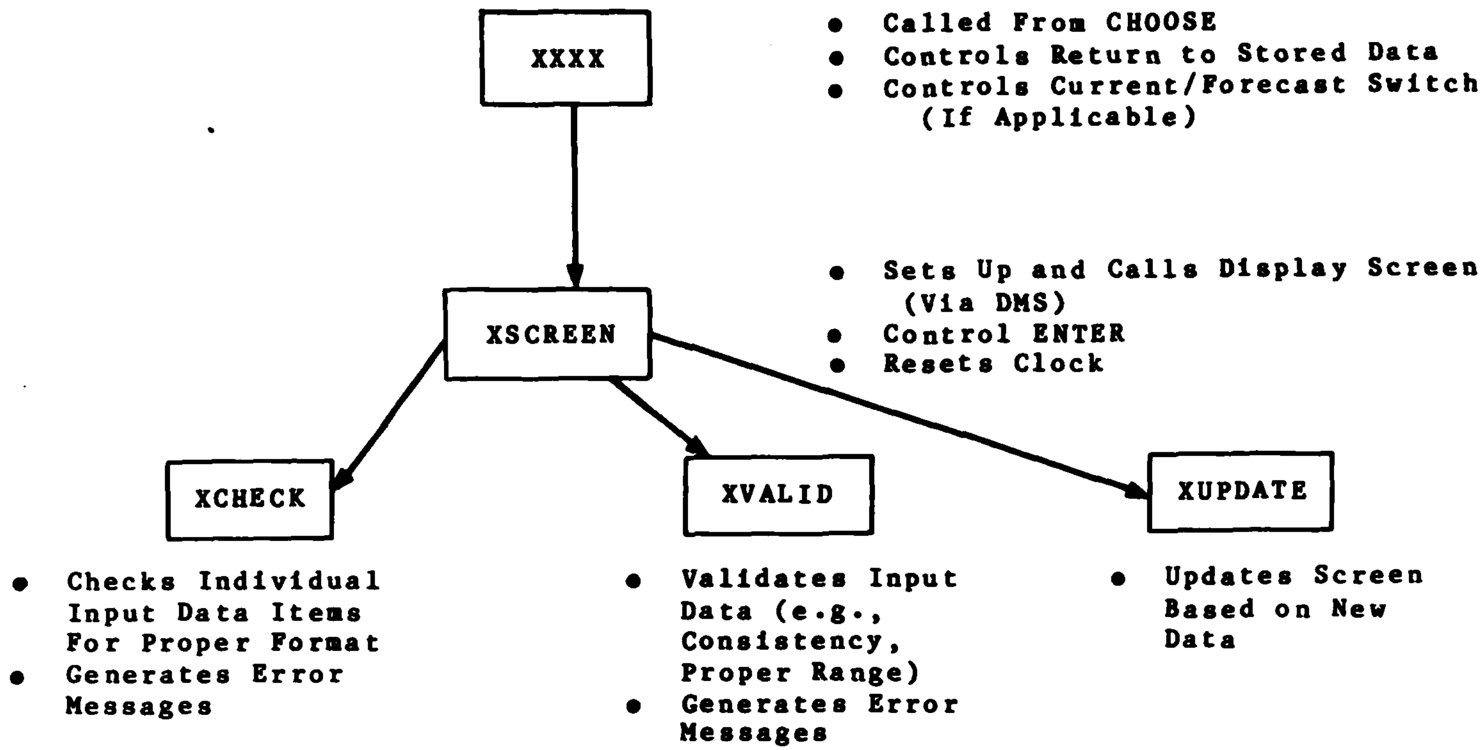


FIGURE 4-3
SCREEN CONTROL FUNCTIONS

TABLE 4-2

SCREEN AVAILABILITY FOR EACH USER

CMS SCREEN	USER AC	USER CAB	USER AF
O'Hare Status	F	NA	NA
Planning Log:			
Weather & Wind	F*	F	NA
Runway Equipment	F*	F	NA
Runway Condition	F*	F	NA
Demand	F	D	NA
Airport Status: Current	F	F	NA
Forecast	F	NA	NA
Runway Equipment Status: Current	F	D	F
Forecast	F	NA	NA
Demand Profile: Current	F	NA	NA
Forecast	F	NA	NA
Ordered List of Configurations: Current	F	NA	NA
Forecast	F	NA	NA
Configuration Information: Current	F	NA	NA
Forecast	F	NA	NA
Menu of PF Keys	F	F	F
Parameters	F	NA	NA
Current Departure Queue	F	F	NA

F - Full Use
NA - Not Available
D - Display Only
* - Indicates Partial Input Capability

5. CMS DATA BASE

All the information essential in describing O'Hare Airport over the planning horizon are stored in a central data base. This data base is housed in an external storage device (disk) and is accessed by each of the three programs based on a given protocol. As the users interact with CMS, the contents of this data base are updated and made available to them.

5.1 Access Mechanism

In order to access the central data base, each program employs a system subroutine COMMD (Reference 7). Subroutine COMMD causes a program interrupt to issue a system command. In this case, the system command issued by each program through subroutine COMMD links the disk storage housing the program to the one containing the CMS data base. Once the link is established the program can proceed by reading or writing data onto and from the data base. After the CMS data base is updated, the subroutine COMMD is called again and a system command is issued releasing the disk containing the data base making it available for access by other users.

5.2 Integrity Mechanism

Since all three programs may access the data base at any time, a system protocol is established disallowing simultaneous access of the data base. If a program issues a link command while the data base is being accessed by another program, the second link is not established and a return code is issued. After receiving a 'not linked' return code, the program issues a wait command via COMMD subroutine. The wait command causes a program interrupt with no action taken for 5 seconds (actual time). After expiration of 5 seconds, the program attempts another link operation. This process continues until the program establishes a successful link to the data base. With this mechanism the simultaneous access of the data base by several users is prevented.

5.3 Data Base Content

The CMS data base contains all information necessary to describe O'Hare Airport over the planning horizon. This central data file is read and written sequentially and contains the following information:

- times when each screen within the CMS package is written (stored) in the data base (character form),

- Midway flag indicator, current operating configuration indicator, and current departure queue lengths (numerical and character form),

- arrival and departure wind thresholds (character and numerical form),

- information on airport status screens (current and forecast) not including those calculated by CMS (character and numerical form),

- information on equipment status screens (current and forecast),

- equipment planning log messages (character and numerical form),

- wind and weather planning log messages (character and numerical form),

- runway conditions planning log messages (character and numerical form),

- demand planning log information (character and numerical form)

The format of the CMS data base is discussed in Appendix E.

5.4 Other CMS Data Files

There are a number of separate data files that contain the information needed by CMS, but are not part of the CMS data base. Since such information are of permanent nature, they are read into the memory prior to the execution of CMS programs. These files are the following:

- File RNWYMIN: contains ceiling and visibility minima associated with each runway with various equipment inoperable.

- File CNFGRQ: contains a list of available configurations with their associated fix assignments and capacity file indices.

- File CAPACTY: contains capacity curves used to compute configuration capacities.

- File TRAVEL: contains fix-to-runway nominal travel times used in transition analysis.

- File DEPEND: contains exclusive dependence matrix used in transition analysis.

- File OAGDMND: contains Official Airline Guide demand profiles used for demand planning.

The formats associated with the above files are discussed in Appendix E.

6. SYSTEM DATA STRUCTURES AND TOP LEVEL PROCESSING

6.1 System Data Structures

The CMS data structures can be categorized in to six distinct groups. In this subsection these groups are discussed. Volume II contains a more detailed description in the form of pseudocodes.

1. Data structures pertaining to information on the CMS permanent data files (i.e., TRAVEL, DEPEND, CAPACTY, etc.).

2. Data structures pertaining to information on the CMS calculated variables (i.e., demand balancing information, configuration capacities, percentage of arrivals, etc.).

3. Data structures pertaining to screen variables: there are two data structures associated with each screen that contain the information on that screen (numerical and character form).

4. Data structures pertaining to current data base information: these data structures contain the data read from or written to the data base.

5. Data structures pertaining to original data base information: these data structures contain a copy of current data base data structures at the time they are read in; used when the data base is being updated to determine what changes have occurred since the last update.

6. PF key variables: associated with each PF key there is a variable identified by DMS and used to invoke various screens.

6.2 Top Level Processing

The logic of the CMS top level processing given in this section is that of the AC program, since it contair- complete collection of the available CMS screens and performs all the functions of CMS. Figure 6-1 shows the high level flow of control of the AC program.

The following logic is used by the AC program:

- initialization that includes reading of the permanent CMS data files and assigning appropriate values to the permanent CMS data structures.

```
Initialization (GETFILE)
DO UNTIL (TERMINATION = 'X')
     Choose Screen or Function (CHOOSE)
     Access Central Airport Data File (TOLINK)
     Read Central Airport Data File (READER)
     Merge all Versions of Central Airport Data File (MERGE)
     Write Central Airport Data File (WRITER)
     Release Central Airport Data File (TODTACH)
     Assign New Values to Non-Computed Variables (ASSIGN)
     Compute Remaining Variables (UPDATE)
END
```

FIGURE 6-1
FLOW OF CONTROL FOR EACH CMS USER PROGRAM

- entering the waiting state by displaying the menu screen and awaiting the user's response.

- interpreting the user's request for new screen, termination, or update.

- entering the execution state.

- choosing a new screen or function.

- reading the current version of data base and setting the original, current, and screen data structures.

- merging different versions of the data base to obtain the most recent one.

- writing the most recent version back onto data base.

- releasing the data base.

- performing a number of analyses and preparing the calculated variables of CMS.

- displaying the new requested screen (waiting state).

CMS makes use of three different versions of the data base: the original version saved at the time when the data base is accessed, the current version which reflects any changes made by the user, and the central data base, which is the latest version, accessed at the time of update and which may contain changes made by other users.

In order to merge different versions of CMS data base contents to obtain the most recent one, the following rule is followed: if the current data base information is not the same as the original copy of the data base saved at the time of last update, then the information obtained from the last screen is the most recent version, otherwise the central data base information is used. Additionally, for computing the calculated variables of CMS the following analyses is performed:

- Compute visibility and ceiling minima based on equipment status.

- Compute crosswind and tailwind components of the wind.

- Determine runway closures.
- Compute north and south demands based on Fix-to-Runway assignments.
- Compute capacity and saturation (demand balancing) for each configuration.
- Update departure runways for current configuration.

7. CMS SCREENS

In this section each screen within the CMS software package is defined and the logic associated with its operation, error checking, and data validation routines are given in form of high and low level pseudocodes.

7.1 Menu and Parameter Screens

The menu display screen is designed to serve two purposes. First, it displays a list of all the available screens and functions and their associated program function keys - analogous to a book's table of contents - to help the user in remembering how to access a particular screen or initiate a specific function. Second, it is used for program termination. Figure 7-1 shows a sample of menu screen for the AC program.

The parameter screen is designed to allow the AC to set the airport's arrival and departure crosswind and tailwind thresholds. These thresholds are used by the AC program to determine when a runway becomes ineligible due to excessive wind. Figure 7-2 contains a sample of parameters screen.

7.2 O'Hare Status Summary Screen

The purpose of the O'Hare status summary display screen is to provide the AC with an overview of the current operating conditions of the airport. For this purpose, the prevailing wind and weather conditions, as well as the current operating configuration and its capacity are displayed. Also shown is the relationship of the capacity for the current runway configuration to the maximum capacity achievable for current conditions.

In addition to the above summary information, the bottom half of the O'Hare status screen contains a number of messages from the wind/weather, airport, and equipment planning logs. These messages serve as a reminder to the AC of what changes are expected, or recently have been made. Figure 7-3 shows a sample of O'Hare status summary screen.

7.3 Planning Log Screens

In order to facilitate the communication among the three users of CMS, a system of interconnected planning log screens are provided. These screens do not affect any of the status screens within the CMS system. Each user of CMS has access to all or a number of the planning log screens, utilizing them to communicate with the other users about the upcoming changes that

```
*****************************************************************************
*                                                                           *
*                          MENU OF PF KEYS                                  *
*                                                                           *
*   -------------------------------------------------------------           *
*   PF  1...O'HARE STATUS SUMMARY                                           *
*   PF  2...PLANNING LOGS                                                   *
*                                                                           *
*   PF  3...AIRPORT STATUS (CURRENT/FORECAST)                               *
*   PF  4...RUNWAY EQUIPMENT STATUS (CURRENT/FORECAST)                      *
*   PF  5...DEMAND PROFILE (CURRENT/FORECAST)                               *
*   PF  6...ORDERED LIST OF CONFIGURATIONS (CURRENT/FORECAST)               *
*   PF  7...CURRENT DEPARTURE QUEUES/ORDERED LIST OF TRANSITIONS            *
*   PF  8...CONFIGURATION INFORMATION (CURRENT/FORECAST)                    *
*                                                                           *
*   PF 10...ACKNOWLEDGE                                                     *
*   PF 11...MENU OF PF KEYS & PROGRAM TERMINATION/PARAMETERS                *
*   PF 12...RETURN TO PREVIOUSLY STORED DATA                                *
*                                                                           *
*   PA  2...HARDCOPY OF DISPLAYED PANEL                                     *
*   --------------------------------------------------------------          *
*                                                                           *
*                                                                           *
*              TO TERMINATE THIS SESSION, ENTER "X"...                      *
*                                                                           *
*                                                                           *
*                                                                           *
*                                                                           *
*****************************************************************************
```

FIGURE 7-1
MENU OF PF KEYS FOR AC

```
PARAMETERS

CROSSWIND THRESHOLDS:

          ARRIVAL............10.0

          DEPARTURE..........15.0


TAILWIND THRESHOLDS:

          ARRIVAL.............10.0

          DEPARTURE...........15.0



DATA STORED AT 1455
```

FIGURE 7-2
PARAMETERS FOR AC

```
O'HARE STATUS

WX:  CEIL... 5000   VIS... 4.50      WIND:  DIR... 060    VEL...3

 ARR...  4R  9R  9L    DEP... 32R 32L        CAPACITY... 204

       CAPACITY AT 94% OF HIGHEST AVAILABLE CAPACITY

                SCROLL...      LINES

               **** RECENT CHANGES ****
1200   22R   HIRL    OTS
1300   WX    5000   4.50   SCATTERED CLOUDS
1300   WIND   060   3      SCATTERED CLOUDS

               **** EXPECTED CHANGES ****
2000   22R   HIRL    RTS
2200   32L   ALS     OTS  REPAIRS
2315   32L   ALS     RTS  REPAIRS



DATA STORED AT 1621
```

FIGURE 7-3
O'HARE STATUS FOR AC

are expected pertaining to runway conditions, weather conditions, equipment status, and demand levels at the airport.

7.3.1 Selection Screen

The purpose of the planning log selection screen is to consolidate the accessing methods of the planning logs that are provided for the AC program. Figure 7-4 contains a sample of planning log selection screen. The planning logs accessed via this screen are:

1. weather and wind planning log
2. airport conditions planning log
3. equipment planning log
4. demand planning log

7.3.2 Weather and Wind Planning Log Screen

The purpose of the weather and wind planning log screen is to serve as a communication tool for both the AC and team supervisor of the tower cab (CAB) in relaying information about the expected changes in the airport's weather and wind conditions. Usually, such information is entered into CMS by the CAB position, and then is used by the AC; however, the AC can enter appropriate messages in the lower half of the screen designated for him/her.

Figure 7-5 shows a sample of weather and wind planning log screen. The types of information entered on this display screen include the time, ceiling, visibility, wind direction, and velocity. Additionally, a remark field is provided for any free formatted comments that the user may deem necessary to include.

7.3.3 Airport Runway Conditions Planning Log Screen

The purpose of the airport runway conditions planning log screen is to serve as a communication tool for both the AC and CAB in relaying information about the runway surface and braking conditions, and runway closures. Usually, such information is entered into CMS by the CAB position, and then is displayed by the AC; however, the AC can enter appropriate messages in the lower half of the screen designated for him.

Figure 7-6 contains a sample of airport runway conditions screen. The types of information entered on this display screen include time, runway ID, surface and/or braking conditions, and runway opening/closures. Additionally, a remark field is

```
**************************************************************
*                                                            *
*                      PLANNING LOGS                         *
*                                                            *
*              (TO SELECT LOG, ENTER "X")                    *
*                                                            *
*          -------------------------------------             *
*          |        | WX & WIND                 |            *
*          |        |                           |            *
*          |        | RUNWAY CONDITIONS         |            *
*          |        |                           |            *
*          |        | RUNWAY EQUIPMENT          |            *
*          |        |                           |            *
*          |        | DEMAND                    |            *
*          -------------------------------------             *
*                                                            *
**************************************************************
```

FIGURE 7-4
PLANNING LOG FOR AC

AIRPORT PLANNING LOG

WEATHER & WIND FORECASTS

GMT	CEIL	VIS	DIR	VEL	REMARKS
1300	500	4.50	060	3	SCATTERED CLOUDS

ASST. CHIEF---ADDITIONAL ENTRIES

1200	800	2.00	090	5	CLOUDY

DATA ENTERED AT 1945

FIGURE 7-5
AIRPORT PLANNING LOG FOR AC
(WEATHER AND WIND FORECASTS)

AIRPORT PLANNING LOG

RUNWAY CONDITIONS

GMT	RWY	SURF	BRAK	CLOSED	OPEN	REMARKS
1700	4R			ARR		

ASST. CHIEF---ADDITIONAL ENTRIES

DATA STORED AT 1506

FIGURE 7-6
AIRPORT PLANNING LOG FOR AC
(RUNWAY CONDITIONS)

provided for any free formatted comments that the user may deem necessary to include.

7.3.4 Equipment Planning Log Screen

The purpose of equipment planning log screen is to serve as a communication tool among the AC, AF, and CAB positions in relaying information about the various equipment outages on different runways at the airport. Usually, such information is entered into CMS by the AF position, and then is displayed by the AC and CAB; however, the AC can enter approprite messages in the lower half of the screen designated for him.

Figure 7-7 shows a sample of equipment planning log screen. The types of information on this display screen include the runway ID, type of equipment, time the outage has occurred or is expected, and the time that that piece of equipment is expected to return to service. Additionally, a remark field is provided for any free formatted comments that the user may deem necessary to include.

7.3.5 Demand Planning Log Screen

In order for the AC to use CMS as a planning tool in selecting new configurations, detailed information about the traffic demand at O'Hare airport is needed. The demand planning log screen is designed to display a set of predetermined demand numbers that are based on the Official Airline Guide (OAG), and historical data for AC's use. The AC has the option to change any of the displayed data depending on the airport's current traffic profile. This screen contains hourly accounts of the total number of arrivals and departures based on the OAG that are expected at O'Hare, and their respective fix distributions. Figure 7-8 contains a sample of demand planning log screen.

7.4 Airport Status Screens

There are two screens associated with the airport status: current and forecast. The purpose of these two screens (current and forecast) is to serve both as communication and planning tools for the AC. The current airport status screen provides the AC with the latest airport information entered and updated by the CAB position, as well as computed by CMS. The AC is also permitted to change or modify any of that information. In contrast, the forecast airport status screen is used only by the AC for planning purposes. The AC can enter the forecast airport condition on this screen, and later use them to determine the favorableness of different configurations in terms of transition and capacity impacts.

RUNWAY EQUIPMENT PLANNING LOG

RWY	EQUIPMENT	OTS	RTS	REMARKS
4L	ALS	1500	1600	REPAIRS

ASST. CHIEF---ADDITIONAL ENTRIES

DATA STORED AT 1447

FIGURE 7-7
EXAMPLE OF CMS SCREEN

DEMAND PLANNING LOG
(TO INITIALIZE LOG, ENTER"X"...)

SCROLL LINES

GMT	TOTALS		ARRIVALS				DEPARTURES			
	ARR	DEP	KUBBS MKE	CGT PLANT	VAINS	FARMM	NORTH MKE	EAST	SOUTH	EAST
1500	72	73	20	25	15	15	13	20	22	18
1600	57	72	10	20	10	17	12	20	22	18
1700	46	43	10	16	10	10	13	10	10	10
1800	72	66	20	10	22	12	18	17	12	19

DATA ENTERED AT 1528

FIGURE 7-8
DEMAND PLANNING LOG FOR AC

Figure 7-9 depicts a sample of airport status screen. The following information is contained on airport status screen:

Information requiring manual inputs (provided by the user)

- airport meteorological conditions (ceiling and visibility)

- airport wind conditions (direction and velocity)

- indication of Midway airport's use of runway 13R under IFR conditions

- tower imposed runway closures

- runway RVR reading

Information provided by CMS (computed)

- runway wind conditions

- runway crosswind and tailwind components

- runway minima for ceiling and visibility

- summary of runway closures due to all possible causes.

7.5 Equipment Status Screens

There are two screens associated with the equipment status: current and forecast. The purpose of these two screens (current and forecast) is to serve both as communication and planning tools for the AC. The current runway equipment status screen provides the AC with the latest status of the equipment on various runways as entered and updated by the AF position. The AC is also permitted to change or modify that information. In contrast, the forecast runway equipment status screen is used only by the AC for the planning purposes. The AC can enter the forecast runway equipment status on this screen, and later use it to determine the favorableness of different configurations in terms of capacity and transition impacts.

Figure 7-10 contains a sample of equipment status screen. The following runway equipment is listed on this screen:

- CAT II
- Localizer (LOC)
- Glide Slope (GS)
- Outer Marker (OM)

CURRENT AIRPORT STATUS

CENTERFIELD: WX: CEIL... 5700 VIS... 5.80
WIND: DIR.... 100 VEL... 3

MIDWAY 13R ARR IN USE... X

RWY	TOWER		SURF	BRK	RVR	WIND				MINIMA		CLOSED	
	ARR	DEP	WET	POOR		DIR	VEL	CRSS	TAIL	CEIL	VIS	ARR	DEP
4R						100	3	3	0	200	.75		
4L						100	3	3	0	402	1.25		
9R						100	3	1	0	200	.50		
9L						100	3	1	0	200	.50		
14R						100	3	2	0	100	.25		
14L						100	3	2	0	100	.25		
22R						100	3	3	2	100	.50		
22L						100	3	3	2	200	.50		
27R						100	3	1	3	200	.50		
27L						100	3	1	3	200	.50		
32R						100	3	2	2	200	.50		
32L						100	3	2	2	200	.50		

DATA STORED AT 1507

FIGURE 7-9
CURRENT AIRPORT STATUS FOR AC

CURRENT RUNWAY EQUIPMENT STATUS
(X INDICATES OUTAGE)

RWY	CAT II	LOC	GS	OM	IM	MM	RAIL	ALS	RVR	HIRL	CL	TDZ	NDB VOR
4R	--				--			X	--		--	--	--
4L	--		--		--	--	--		--		--	--	
9R	--				--							--	
9L	--				--						--	--	--
14R													
14L													
22R	--				--				--		--	--	
22L	--				--				--		--	--	--
27R	--				--						--	--	
27L	--				--							--	--
32R	--				--								
32L	--				--								

DATA STORED AT 1500

FIGURE 7-10
CURRENT RUNWAY EQUIPMENT STATUS FOR AC

- Inner Marker (IM)
- Middle Marker (MM)
- Runway Alignment Indicator Lights (RAIL)
- Approach Lighting System (ALS)
- High Intensity Runway Lights (HIRL)
- Runway Visual Range (RVR)
- Centerline Lights (CL)
- Touch Down Zone (TDZ)
- Non-Directional Beacon/VHF Omni-Directional Range (NDB/VOR)

7.6 Demand Profile Screens

There are two screens associated with the demand profile: current and forecast. The purpose of these two screens (current and forecast) is to serve as planning tools for the AC. In order for CMS to compute capacities of different configurations and analyze the transitions, the total hourly traffic demand and its distribution at the fixes for both the arrivals and departures are needed. The demand profile display screens are designed to provide such information to CMS. Figure 7-11 shows a sample of demand profile screen.

7.7 Ordered List of Configurations Screens

The purpose of these two screens (current and forecast) is to serve as planning tools for the AC. The current and forecast ordered list of configurations display screens provide the AC with a list of all the eligible configurations ranked by their respective capacities under specified airport, equipment, and demand conditions. The AC can then use this information for selecting the next best configuration. Both screens are identical in format; each contains the airport's percentage of arrivals and number of eligible configurations for its corresponding environment, Additionally, each screen provides a list of configurations with their respective capacities. Figure 7-12 shows a sample of ordered list of configurations screen.

7.8 Current Departure Queue Screen

Since the departure queues at O'Hare have a substantial impact on configuration changes, CMS requires them for its transition analysis. Every time the AC wishes to use CMS to assess the favorableness of various transitions, the length of departure queues (number of aircraft) for current departure runways need to be supplied. Figure 7-13 contains a sample of departure queue length screen.

```
CURRENT DEMAND  (FROM 2534 TO 1634)

RETRIEVE...

      ARRIVALS:

TOTAL...   64

KUBBS...   14
CGT.....   22
VAINS...   12
FARMM...   16

      DEPARTURES:

TOTAL...   72

NORTH...   12
EAST....   20
SOUTH...   22
WEST....   18


DATA ENTERED AT 1534
```

FIGURE 7-11
CURRENT DEMAND FOR AC

CURRENT ORDERED LIST OF CONFIGURATIONS

TOTAL ARRIVALS... 47%

NUMBER OF ELIGIBLE CONFIGURATIONS... 73

SCROLL LINES

RANK	ARRIVALS	DEPARTURES	CAPACITY	REMARKS
1	22R 27R 27L	22L 32L	225	MIDWAY
2	4R 9R 9L	4L 32L	216	MIDWAY
3	9R 14R 22R	9L 32L	213	MIDWAY
4	4R 9R 9L	32R 32L	208	MIDWAY
5	9R 14R 22R	14L 22L	200	MIDWAY
6	9R 14R 14L	4L 22L	199	MIDWAY
7	9R 14R 14L	9L 22L	192	MIDWAY
8	9R 14R 14L	9R 22L	191	MIDWAY
9	14R 14L 22L	22L 27L	190	MIDWAY
10	9R 14R 14L	4R 4L	188	

DATA STORED AT 1534

FIGURE 7-12
CURRENT ORDERED LIST OF CONFIGURATIONS FOR AC

CURRENT DEPARTURE QUEUES

DEPARTURE RUNWAYS	QUEUE LENGTH
4R	1
4L	3

DATA ENTERED AT 1536

FIGURE 7-13
CURRENT DEPARTURE QUEUES FOR AC

7.9 Ordered List of Transitions Screen

The purpose of this screen is to serve as a planning tool for the AC. The ordered list of transitions display screen provides the AC with a list of all the possible transitions ranked based on their respective transition hour capacities. The AC can then use this information in selecting the next configuration. This screen contains the airport's forecast percentage of arrivals and number of eligible configurations. Additionally, the screen contains the airport's current operating configuration and a list of configurations eligible for transition. Also, for each such configuration, a theoretical transition duration, a transition hour capacity, and a final steady state capacity are given. Figure 7-14 contains a sample of ordered list of transitions screen.

7.10 Configuration Information Screens

There are two screens associated with the configuration information: current and forecast. The purpose of these two screens (current and forecast) is to serve as planning tools for the AC. If the AC wishes to acquire detailed information on the performance of any eligible configuration under current or forecast conditions, he can use these screens.

Figure 7-15 contains a sample of configuration information screen. These screens contain the following information for the north complex, south complex, and entire airport:

1. percentage of arrivals
2. saturation levels
3. arrival and departure demands
4. arrival and departure capacities.

Also, included in these screens is an area where a new configuration can be entered, if the AC wishes to review its detailed information.

ORDERED LIST OF TRANSITIONS

ARRIVALS... 47%   NUMBER OF ELIGIBLE CONFIGURATIONS... 73

SCROLL   LINES

RANK	ARRIVALS	DEPARTURES	TRANSITION DUR (MIN)	TRANSITION HOUR CAP	FINAL CAP
CURRENT	9R 9L	4R 4L	--	155	154
1	4R 9R 9L	4L 32L	28	213	216
2	22R 27R 27L	32L 32L	28	209	226
3	4R 9R 9L	32R 32L	28	207	208
4	9R 14R 22R	9L 22L	28	204	214
5	9R 14R 22R	14L 22L	28	194	201
6	9R 14R 14L	4L 22L	28	191	199
7	9R 14R 14L	4R 4L	28	186	188
8	9R 14R 14L	9L 22L	28	186	191
9	9R 14R 14L	9R 22L	28	185	190
10	4R 9R 9L	4L 9R	28	181	179

DATA STORED AT 1534

FIGURE 7-14
ORDERED LIST OF TRANSITIONS FOR AC

CURRENT CONFIGURATION

	4R	4L	9R	9L	14R	14L	22R	22L	27R	27L	32R	32L
ARRIVALS							X	X				
DEPARTURES										X	X	X

	PCT ARR	SAT	ARRIVAL DEM	ARRIVAL CAP	DEPARTURE DEM	DEPARTURE CAP
TOTAL	46%	.92	53	57	63	68
NORTH	43%	.91	26	28	34	37
SOUTH	48%	.93	27	29	29	31

5 ARRIVALS MOVED TO NORTH COMPLEX
5 DEPARTURES MOVED TO NORTH COMPLEX

DATA STORED AT 1336

FIGURE 7-15
CURRENT CONFIGURATION FOR AC

8. INTERCONNECTIVITY OF SCREENS

In this section the relationships betwen various screens within the CMS software package are discussed. Although each CMS screen is designed to serve a specific purpose, they are not mutually exclusive (i.e., there may be an overlap of information between two or more screens). There are two types of relationships that may exist between two or more screens: implicit and explicit.

Although two or more screens may not have an overlap of data, changes in one screen might directly affect other screens. This type of connectivity between screens is defined as implicit connectivity. For example, a change in the values of tailwind and crosswind thresholds on parameter screen may affect the results of runway availability analysis on airport status screen.

On the other hand, the explicit relationship among CMS screens is defined as simply the overlap of the same information between two or more screens. For example, the prevailing ceiling is reflected on O'Hare status screen as well as current airport status screen. Figure 8-1 and Table 8-1 depict the implicit and explicit connectivity between various CMS screens.

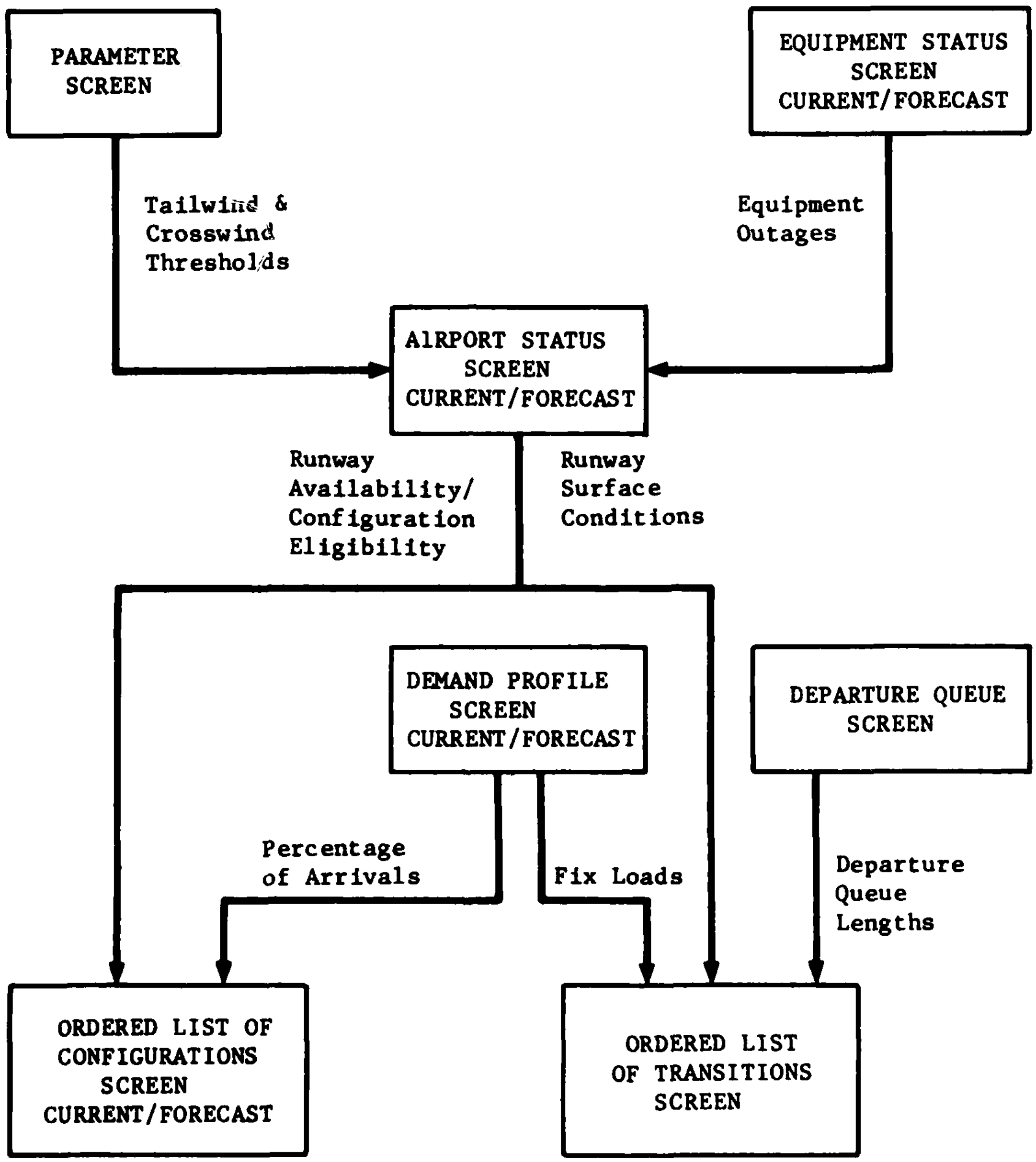


FIGURE 8-1
IMPLICIT CONNECTIVITY OF CMS SCREENS

TABLE 8-1
EXPLICIT CONNECTIVITY OF CMS SCREENS

	O'HARE STATUS	AIRPORT STATUS		DEMAND PROFILE		ORDERED LIST OF CONFIGURATIONS		ORDERED LIST OF TRANSITIONS	CONFIGURATION INFORMATION		PLANNING LOGS		
		CRNT	FCST	CRNT	FCST	CRNT	FCST		CRNT	FCST	WX/WIND	EQUIPMENT	RUNWAY SURFACE
Prevailing Ceiling/Visibility	X	X											
Prevailing Wind Direction/Velocity	X	X											
Operating Configuration	X					X		X	X				
Operating Configuration Capacity	X					X		X	X				
Synopsis of Various Log Messages	X										X	X	X
Eligible Configuration's Capacities						X	X		X	X			
Percentage of Arrivals						X	X	X	X	X			
Arrival/Departure Demands				X	X				X	X			

APPENDIX A

CMS SOFTWARE LOGIC - HIGH LEVEL PSEUDOCODES

This appendix presents the CMS software in the form of high level pseudocodes that describe the purpose and logic of each CMS routine, irrespective of its variable specifications.

A more detailed description of the CMS software logic is given in Volume II (low level pseudocodes). Additionally, a cross-reference table (Table A-1), where the CMS software routines are listed alphabetically with their locations within Appendix A and Volume II, is given. The high level pseudocodes are divided into the following modules:

TABLE A-1

CROSS REFERENCE OF HIGH LEVEL AND LOW LEVEL PSEUDOCODES FOR APPENDIX A AND VOLUME II

Routine	High Level (Appendix A)	Low Level (Volume II)
ACHECK	A-116	2-272
ADJST	A-162	2-389
AGLOBAL	A-21	2-82
ARPT	A-112	2-267
ASCREEN	A-113	2-268
ASSIGN	A-20	2-79
AUPDATE	A-118	2-278
AVALID	A-118	2-276
CALC	A-160	2-371
CAPCAL	A-45	2-131
CAPSAT	A-39	2-120
CCHECK	A-173	2-417
CGLOBAL	A-23	2-91
CHOOSE	A-11	2-66
CLOSING	A-30	2-96
CNFG	A-165	2-401
CONSET	A-158	2-361
CSCREEN	A-166	2-402
CUPDATE	A-173	2-420
CVALID	A-173	2-419
DBAL	A-46	2-133
DCHECK	A-129	2-298
DEMSET	A-158	2-362
DGLOBAL	A-22	2-87
DMND	A-125	2-293
DSCREEN	A-126	2-294
DUR	A-162	2-385
DVALID	A-133	2-306
ECHECK	A-93	2-233
EDP	A-163	2-391
EGLOBAL	A-23	2-86
ELIG	A-31	2-104
ELOG	A-89	2-229
ESCREEN	A-90	2-230
EUPDATE	A-100	2-244
EVALID	A-95	2-237
FILES	A-36	2-115
GCHECK	A-105	2-252
GETFILE	A-7	2-61
GGLOBAL	A-25	2-95
GLOG	A-102	2-248

TABLE A-1
(Continued)

Routine	High Level	Low Level
GSCREEN	A-103	2-249
GVALID	A-110	2-260
HCHECK	A-58	2-176
HSCREEN	A-54	2-166
HSTAT	A-49	2-152
INREAD	A-9	2-63
LCHECK	A-63	2-183
LOGS	A-60	2-179
LSCREEN	A-60	2-180
LUPDATE	A-64	2-185
LVALID	A-64	2-174
MENUPRM	A-175	2-424
MERGE	A-16	2-73
MILES	A-8	2-61
MINIMA	A-26	2-99
MSCREEN	A-176	2-426
OBJFUN	A-163	2-396
OCHECK	A-143	2-328
ORDER	A-136	2-313
OSCREEN	A-140	2-321
OSETUP	A-137	2-315
OSORT	A-139	2-319
OUPDATE	A-143	2-331
OVALID	A-143	2-330
PCHECK	A-180	2-430
PERCENT	A-37	2-116
PGLOBAL	A-24	2-91
PSCREEN	A-177	2-427
PVALID	A-180	2-432
QCHECK	A-149	2-338
QFIX	A-38	2-119
QGLOBAL	A-24	2-93
QSCREEN	A-146	2-335
QUEUE	A-145	2-334
QVALID	A-149	2-339
RCHECK	A-123	2-285
READER	A-14	2-71
RGLOBAL	A-22	2-84
RHO	A-47	2-146
RSCREEN	A-121	2-282
RUPDATE	A-123	2-288
RWY	A-120	2-281

TABLE A-1
(Concluded)

Routine	High Level	Low Level
SCHECK	A-81	2-213
SGLOBAL	A-25	2-94
SLOG	A-77	2-209
SPTRAN	A-159	2-366
SSCREEN	A-78	2-209
SUPDATE	A-87	2-225
SVALID	A-83	2-217
TCHECK	A-154	2-349
TDEP	A-159	2-364
TODTACH	A-14	2-71
TOLINK	A-14	2-70
TRAN	A-155	2-350
TSCREEN	A-152	2-343
TSETUP	A-151	2-342
UPDATE	A-21	2-80
WCHECK	A-69	2-192
WGLOBAL	A-25	2-93
WIND	A-30	2-97
WLOG	A-66	2-188
WRITER	A-15	2-72
WSCREEN	A-67	2-189
WUPDATE	A-75	2-205
WVALID	A-72	2-197

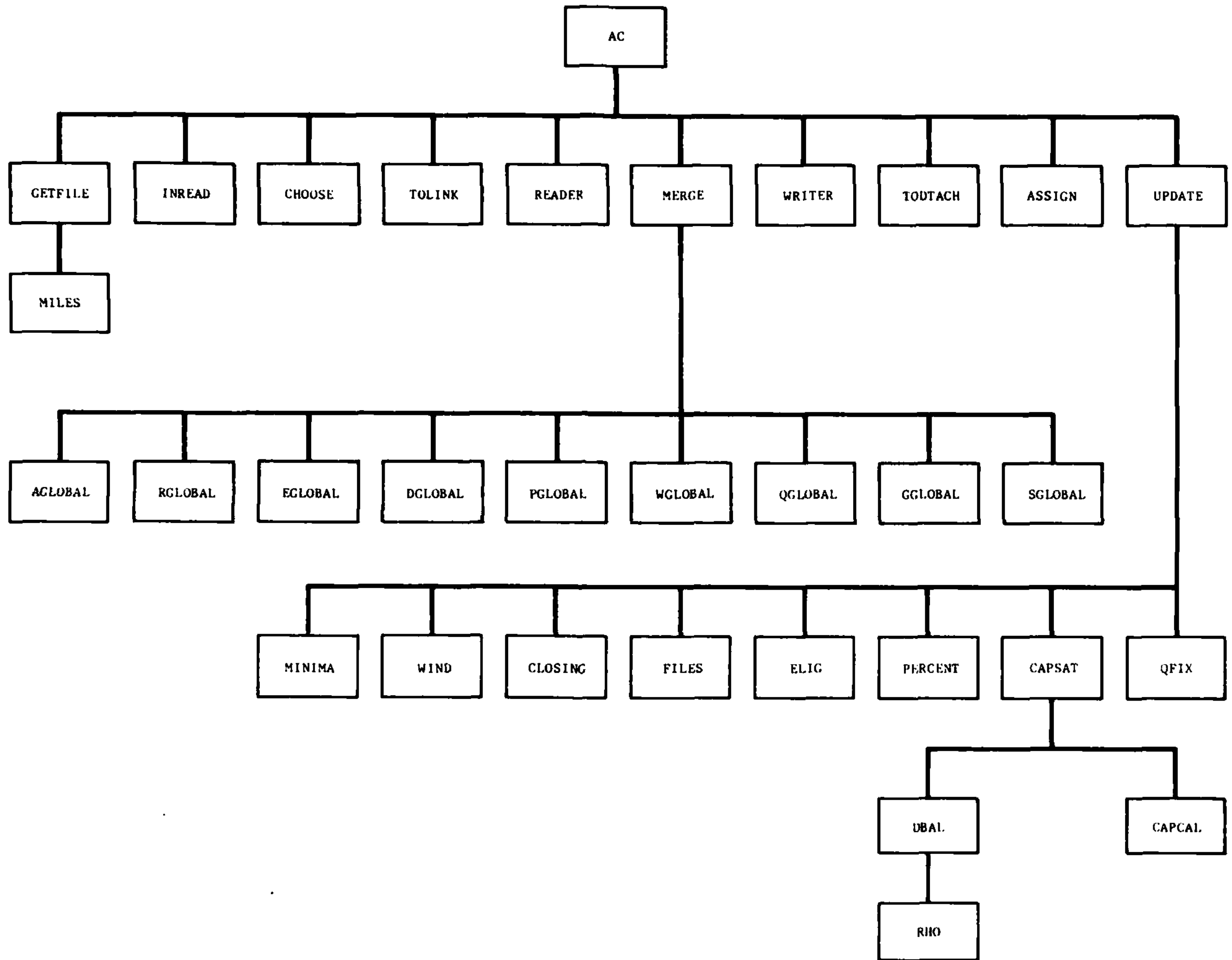


FIGURE A-1
SCHEMATIC DIAGRAM OF
TOP LEVEL PROCESSING ROUTINES

```
TASK  ASSISTANT_CHIEF_MAIN_PROGRAM
    [This is assistant chief program's main procedure referred to as ASSISTANT_CHIEF_MAIN_PROGRAM, it
    controls entire program by calling several routines that take the user into CMS]

CALL  GETFILE;  [read permanent data files containing program's global parameters]

CALL  INREAD;  [link to data base initially, read data base]

REPEAT UNTIL  (termination indicator is set);

    CALL  CHOOSE;  [choose screen or function]

    CALL TOLINK;   [access central data base]

    CALL READER; (read central data base)

    CALL MERGE;  [merge all versions of data base to obtain most recent version]

    CALL WRITER;   [write most recent version onto data base]

    CALL TODTACH;  [release central data base]

    CALL ASSIGN;   [assign most current version to internal variables preparang for next cycle]

    CALL UPDATE;   [update CMS computed variables]

ENDREPEAT;

END  ASSISTANT_CHEIF_MAIN_PROGRAM;
```

```
ROUTINE  GETFILE
    [This routine reads permanent data files containing CMS global parameters.]

    Read runway minima parameters;

    CALL  MILES; [convert runway visibility minima parameters from RVR readings to miles]

    Read configuration information parameters;
    Read capacity file;
    Read travel time parameters;
    Read dependence matrix parameters;
    Read Official Airlines Guide demand information;

END  GETFILE;
```

```
ROUTINE  MILES
         [This routine converts RVR readings to miles]

   REPEAT;   [for each runway]

       PERFORM   RVR_TO_MILES_CONVERSION;

   ENDREPEAT;

END  MILES;


PROCESS  RVR_TO_MILES_CONVERSION
   [This process converts each runway visibility minimum data item from RVR to miles using FUNCTION M]

END  RVR_TO_MILES_CONVERSION;
```

```
ROUTINE  INREAD
    [This routine accesses and reads data base initially]

    CALL  TOLINK;  [link to central data base]

    PERFORM  READ_DATA_BASE;  [read all variables from data base]

    CALL  TODATCH; [release central data base]

    PERFORM  SET_ORIGINAL_PROGRAM_VARIABLES;     [establish a copy of recently read program variables to be
                                                  original variables]

    PERFORM  SET_CMS_PROGRAM VARIABLES;     [establish a copy of recently read program variables to be used
                                             by CMS lower level programs]

END INREAD;


PROCESS  READ_DATA_BASE
    [This process reads all variables from data base]

    Open STARTUP file and read following variables:

        STORED, MNON, CNOW, QNOW, CVTQNOW, PNOW, CVTPNOW, ANOW, CVTANOW, RNOW, DNOW, CVTDNOW, ENOW, CVTENOW,
        WNOW, CVTWNOW, SNOW, CVTSNOW, GNOW, CVTGNOW;

    Close STARTUP file;

END READ_DATA_BASE;


PROCESS  SET_ORIGINAL_PROGRAM_VARIABLES;
    [This process establishes a copy of recently read program variables to be original variables.]

    Following variables are established:

    MBEGIN, PBEGIN, CVTPBGN, ABEGIN, CVTABGN, RBEGIN, DBEGIN, CVTDBGN, CBEGIN, EBEGIN, CVTEBGN, QBEGIN,
    CVTQBGN, WBEGIN, CVTWBGN, SBEGIN, CVTSBGN, QGEGIN, CVTGBGN;

END SET_ORIGINAL_PROGRAM_VARIABLES;
```

```
PROCESS  SET_CMS_PROGRAM_VARIABLES;
    {This process establishes a copy of recently read program variables to be used by CMS lower level
    programs}

    Following variables are established:

    MIDFLAG, PARAM, CNVTPRM, APTSTAT, CNVTAPT, RWYEQP, DEMAND, CNVTDEM, CONFIND, EQPLOG, CNVTEQP, QUELEN,
    CNVTQLN, WXLOG, CNVTWX, SURFLOG, CNVTSRF, OAGLOG, CNVTOAG;

END SET_CMS_PROGRAM_VARIABLES;
```

```
ROUTINE  CHOOSE
    [This routine checks value of current program status variable and chooses function or screen desired by
    CMS user]

    IF  current program status is set to acknowledge function (PF10)

        THEN  set current program status to previous program status;

        ELSE  set previous program status to current program status;

    PERFORM   SCREEN_SELECTION;

END CHOOSE;
```

```
PROCESS  SCREEN_SELECTION
   [This process selects current screen]

   IF  current program status is set to O'Hare status screen (PF1)

       THEN CALL HSTAT;    [O'Hare status screen]

       ELSEIF    current program status is set to log selection screen (PF2)

           THEN CALL  LOGS [log selection screen]

           ELSEIF    current program status is set to wind and weather planning log screen (PF13)

               THEN CALL  WLOG; [wind and weather planning log screen]

               ELSEIF    current program status is set to airport planning log screen (PF14)

                   THEN CALL  SLOG;    [airport planning log screen]

                   ELSEIF  current program status is set to equipment planning log screen (PF15)

                       THEN CALL ELOG; [equipment planning log screen]

                       ELSEIF  current program status is set to demand planning log screen (PF16)

                           THEN CALL GLOG; [demand planning log screen]

                           ELSEIF  current program status is set to airport status screen (PF3)

                               THEN CALL ARPT;  [airport status screen (current, forecast)]

                               ELSEIF    current program status is set to equipment status screen
                                         (PF4)

                                   THEN CALL RWY; [equipment status screen (current, forecast)]

                                   ELSEIF    current program status is set to ordered list of
                                             configurations screen (PF6)
```

```
THEN CALL ORDER;    [order list of configurations screen
                     (current, forecast)]

ELSEIF    current program status is set to current departure
          queue length screen (PF7)

     THEN CALL QUEUE;    [current departure queue length
                          screen]

     ELSEIF    current program status is set to ordered list
               of transitions screen (PF8)

          THEN CALL TSETUP;    [current ordered list of
                                transitions screen]

          ELSEIF    current program status is set to
                    configuration information screen (PF9)

               THEN CALL CNFG;     [configuration
                                    information screen
                                    (current, forecast)]

               ELSEIF    current program status is set to
                         acknowledge (PF10)

                    THEN [previous program status is set to
                         menu screen (PF11)]

                    ELSE CALL MENUPRM;
                         [menu screen or parameter screen]

END  SCREEN_SELECTION;
```

```
ROUTINE  TOLINK
   [This routine establishes a link to data base]

   CALL COMMD; [a system routine that issues system messages, in this instance a link command is issued]

   IF   return code is greater than 106 AND is less than 120

        THEN  stop; [a linkage error has occurred]

   REPEAT WHILE  (return code is not equal to 0, linkage is not established due to linkage from other user);

        CALL COMMD; [a wait command is issued]

        CALL COMMD; [a link command is issued]

   ENDREPEAT;

   CALL COMMD;    [issue access command once linked]

END TOLINK;


ROUTINE  TODTACH
   [This routine detaches user from data base]

   CALL COMMD; [issue a detach command]

   CALL COMMD; [issue a release command]

END TODTACH;

ROUTINE  READER
   [This routine reads data base into current global variables]

   PERFORM  READ_DATA_BASE; [read all variables from data base]

END READER;
```

```
ROUTINE  WRITER
    [This routine writes most current version of data on to data base]

        Open STARTUP file and write following variables: STORED, MNOW, CNOW, QNOW, CVTQNOW, PNOW, CVTPNOW,
        ANOW, CVTANOW, RNOW, DNOW, CVTDNOW, ENOW, CVTENOW, WNOW, CVTWNOW, SNOW, CVTSNOW, GNOW, CVTGNOW:

        Close STARTUP file;

END  WRITER;
```

```
ROUTINE MERGE
    [This routine merges and reconciles all different versions of data base to obtain most current version, a
    number of routines that perform global updates are called from this routine]

    IF  previous program status is set to O'Hare status screen (PF1) AND O'Hare status screen message is not
        equal to 'DATA STORED'

        THEN  update time on STORED variable pertaining to O'Hare status screen;

        ELSEIF    previous program status is set to wind and weather planning log screen (PF13) AND wind and
                  weather planning log screen message is not equal to 'DATA STORED'

            THEN

                CALL WGLOBAL; [reconcile different versions of wind and weather planning log information]

                Update time on STORED variable pertaining to wind and weather planning log and O'Hare
                status screens;

            ELSEIF    previous program status is set to airport planning log screen (PF14) AND airport
                      planning log screen message is not equal to 'DATA STORED'

                THEN

                    CALL SGLOBAL; [reconcile different versions of airport planning log information]

                    Update time on STORED variable pertaining to runway conditions planning log and
                    O'Hare status screens;

                ELSEIF    previous program status is set to equipment planning log screen (PF15) AND
                          equipment planning log screen message is not equal to 'DATA STORED'

                    THEN

                        CALL EGLOBAL; [reconcile different versions of equipment planning log
                            information]

                        Update time on STORED variable pertaining to equipment planning log and O'Hare
                        status screens;

                    ELSEIF    previous program status is set to demand planning log screen (PF16) AND
                              demand planning log screen message is not equal to 'DATA STORED'

                        THEN
```

```
CALL GGLOBAL; [reconcile different versions of demand planning log
     information]

Update time on STORED variable pertaining to demand planning log screen;

ELSEIF   previous program status is set to airport status screen (PF3) AND
         airport status screen message for either operating environment is not
         equal to 'DATA STORED'

THEN

      CALL AGLOBAL; [reconcile different versions of airport status
           information]

      Update time on STORED variable pertaining to airport status, ordered
      lists of configurations, configuration information, and ordered list
      of transitions screens;

      ELSEIF   previous program status is set to equipment status screen (PF4)
               AND equipment status screen message for either operating
               environment is not equal to 'DATA STORED'

      THEN

            CALL RGLOBAL;  [reconcile different versions of equipment status
                           information]

            Update time on STORED variable pertaining to equipment status,
            ordered lists of configurations, configuration information, and
            ordered list of transitions screens;

            ELSEIF   previous program status is set to demand profile screen
                     (PF5) AND demand profile screen message for either operating
                     environment is not equal to 'DATA STORED'

            THEN

                  CALL DGLOBAL; [reconcile different versions of demand profile
                       information]

                  Update time on STORED variable pertaining to demand profile,
                  ordered lists of configurations, configuration information, and
                  ordered list of transitions screens;
```

```
ELSEIF    previous program status is set to ordered list of
          configurations (PF6) AND ordered list of configurations
          screen message for either operating environment is not
          equal to 'DATA STORED'

     THEN

          CALL CGLOBAL;  [reconcile different versions of
                         configuration information]

          IF   operating environment is equal to current

               THEN update time on STORED variable pertaining to
                    ordered lists of configurations,
                    configuration information, departure queue
                    length, and ordered list of transitions
                    screens;

               ELSE update time on STORED variable pertaining to
                    ordered list of configurations and
                    configuration information screens

     ELSEIF    previous program status is set to current
               departure queue screen (PF7) AND current departure
               queue screen message is not equal to 'DATA STORED'

          THEN

               CALL QGLOBAL;  [reconcile different versions of
                              departure queue information]

               Update time on STORED variable pertaining to
               departure queue and ordered lists of transitions
               screens;

          ELSEIF    previous program status is set to ordered
                    list of transitions screen (PF8) AND ordered
                    list of transitions screen message is not
                    equal to 'DATA STORED'

          THEN update time on STORED variable pertaining to
               ordered list of transitions screen;
```

```
ELSEIF   previous program status is set to
         configuration information screen (PF9) AND
         configuration information screen message for
         either operating environment is not equal to
         'DATA STORED'

   THEN

         CALL CGLOBAL; [reconcile different versions
              of configuration information]

         IF   operating environment is equal to current

              THEN update time on STORED variable
                   pertaining to departure queue and
                   ordered lists of transitions
                   screens;

              ELSE update time on STORED variable
                   pertaining to ordered list of
                   configurations and configuration
                   information screens;

      ELSEIF   previous program status is set to
               menu/parameter screen (PF11) AND
               parameter screen message is not equal to
               'DATA STORED'

         THEN

               CALL PGLOBAL; [reconcile different
                    versions of parameter information]

               Update time on STORED variable
               pertaining to parameter, airport status,
               ordered list of configurations,
               configuration information, and ordered
               lists of transitions screens;

END MERGE;
```

```
ROUTINE ASSIGN
    [This routine produces two copies of global variables, one to be used in lower level programs, and other
    to serve as original version until next update cycle]

    PERFORM  SET_ORIGINAL_PROGRAM_VARIABLES;  [establish a copy of recently read program variables to be
                                               original variables]

    PERFORM  SET_CMS_PROGRAM_VARIABLES;   [establish a copy of recently read program variables to be used
                                           by CMS lower level programs]

    PERFORM  STORED_TIME_SET_UP: [set up message portion of global variables with stroed times]

END ASSIGN;


PROCESS  STORED_TIME_SET_UP
    [This process sets up message portion of global variables with stored times]

    Set all screen messages equal to 'DATA STORED' concatenated with stored times pertaining to each screen
    from STORED variable;

END STORED_TIME_SET_UP;
```

```
ROUTINE  UPDATE
    [This routine performs a number of inner model computations needed during each update cycle, e.g.,
    weather minima, crosswind and tailwind components of wind, runway closures, and configuration
    eligibility, etc.]

    REPEAT (for each operating environment);

        CALL MINIMA; [compute minima based on equipment status]

        CALL WIND;     [compute crosswind and tailwind components of wind for each runway]

        CALL CLOSING;  [determine runway closures]

        CALL FILES;    [determine capacity file number for each configuration and set CNDTN variable to
                       indicate VFR(=1) or IFR(=2)]

        CALL ELIG:     [determine eligibility of configurations]

        CALL PERCENT; [compute north and south demands based on fix-to-runway assignments]

        CALL CAPSAT;   [compute capacity and balance demand for each eligible configuration]

    ENDREPEAT;

    CALL QFIX;     [update departure runways for current configuration]

END UPDATE;


ROUTINE  AGLOBAL
    [This routine reconciles different versions of airport status information]

    REPEAT;   (for each runway)

    IF   current CMS data is not equal to original CMS data for each input data field (both character and
         numerical if applicable)

         THEN set central file data equal to current CMS data;

         ELSE set current CMS data equal to central file data;

    ENDREPEAT;

END AGLOBAL;
```

```
ROUTINE  RGLOBAL
    [This routine reconciles different versions of equipment status information]

    REPEAT;   (for each runway)

    IF   current CMS data is not equal to original CMS data for each input data field (character)

             THEN set central file data equal to current CMS data;

             ELSE set current CMS data equal to central file data;

    ENDREPEAT;

END RGLOBAL;


ROUTINE  DGLOBAL
    [This routine reconciles different versions of demand profile information]

    IF   current CMS data is not equal to original CMS data for each input data field (both character and
         numerical)

         THEN set central file data equal to current CMS data;

         ELSE set current CMS data equal to central file data;

END DGLOBAL;
```

```
ROUTINE EGLOBAL
    [This routine reconciles different versions of equipment planning log]

    REPEAT;   (for each message designated to AC use)

        Set central file data equal to current CMS data for each input data field (both character and
        numerical if applicable)

    ENDREPEAT;

END EGLOBAL;


ROUTINE CGLOBAL
    [This routine reconciles different versions of operating configurations information]

    IF   current CMS version of ope-ating configuration's index is not equal to original CMS version of
         operating configuration's index

         THEN

              Set central data file version of operating configuration's index to current CMS version of
              operating configuration's index;

              IF  environment indicator is equal to current environment

                  THEN for all departure runways in new operating configuration set departure queue lengths
                       (character and numerical) equal to zero;

         ELSE set current CMS version of operating configuration's index equal to central data file version
              of operating configuration's index;

END CGLOBAL;
```

```
ROUTINE  QGLOBAL
    [This routine reconciles different versions of current departure queue information]

    REPEAT; (for each departure runway in current operating configuration)

        IF   current CMS data is not equal to original CMS data

             THEN  set central file data (character and numerical) equal to current CMS data;

             ELSE  set current CMS data (character and numerical) equal to central file data;

    ENDREPEAT;

END  QGLOBAL;


ROUTINE  PGLOBAL
    [This routine reconciles different versions of parameters information]

    IF   current CMS data is not equal to original CMS data for each input data field (both character and
         numerical)

         THEN  set central file data equal to current CMS data;

         ELSE  set current CMS data equal to central file data;

END  PGLOBAL;
```

```
ROUTINE  SGLOBAL
    [This routine reconciles different versions of airport planning log information]

    REPEAT; (for each message designated to AC use)

        Set central file data equal to current CMS data for each input data field (both character and
        numerical if applicable);

    ENDREPEAT;

END  SGLOBAL;


ROUTINE  WGLOBAL
    [This routine reconciles different versions of wind and weather planning log information]

    REPEAT; (for each message designated to AC use)

        Set central file data equal to current CMS data for each input data field (both character and
        numerical if applicable);

    ENDREPEAT;

END  WGLOBAL;


ROUTINE  GGLOBAL
    [This routine reconciles different versions of demand planning log information]

    REPEAT; (for each hour)

        Set central file data equal to current CMS data for each input data field (both character and
        numerical);

    ENDREPEAT;

END  GGLOBAL;
```

```
ROUTINE MINIMA
   [This routine computes ceiling and visibility minima based on existing airport's equipment status]

   REPEAT;   (for each runway)

       IF  CATII is operable

          THEN
              Set runway ceiling minimum equal to ceiling minimum with CATII operable;

              Set visibility minimum equal to visibility minimum with CATII operable;

              Convert runway ceiling and visibility minima to character form;

          ELSE  [CATII is inoperable]

              IF  localizer and NDB_VOR are inoperable

                  THEN

                      Set runway ceiling and visibility minima to an arbitrary large number;

                      Set character form of runway ceiling and visibility minima to blanks;

                  ELSEIF  localizer is inoperable AND NDB_VOR is operable

                      THEN

                          Set runway ceiling minimum equal to ceiling minimum with localizer
                          inoperable and NDB_VOR operable;

                          Set runway visibility minimum equal to visibility minimum with localizer
                          inoperable and NDB_VOR operable;

                          IF  RAIL is inoperable

                              THEN

                                  Set runway visibility minimum equal to maximum of runway
                                  visibility minimum and runway visibility minimum with NDB_VOR
                                  operable and localizer and RAIL inoperable;
```

Set runway ceiling minimum equal to maximum of runway ceiling minimum and runway ceiling minimum with NDB_VOR operable and localizer and RAIL inoperable;

Convert runway ceiling and visibility minima to character form;

<u>ELSEIF</u> localizer is operable <u>AND</u> glide slope is inoperable

<u>THEN</u>

Set runway ceiling minimum equal to ceiling minimum with localizer operable and glide slope inoperable;

Set runway visibility minimum equal to visibility minimum with localizer operable and glide inoperable;

<u>IF</u> middle marker is inoperable

<u>THEN</u>

Set runway visibility minimum equal to maximum of runway visibility minimum and runway visibility minimum with localizer operable and glide slope and middle marker inoperable;

Set runway ceiling minimum equal to maximum of runway ceiling minimum and runway ceiling minimum with localizer operable and glide slope and middle marker inoperable;

<u>IF</u> ALS is inoperable

<u>THEN</u>

Set runway visibility minimum equal to maximum of runway visibility minimum and runway visibility minimum with localizer operable and glide slope and ALS inoperable;

Set runway ceiling maximum equal to maximum of runway ceiling minimum and runway ceiling minimum with localizer operable and glide slope and ALS inoperable;

Convert runway ceiling and visibility minima to character form;

ELSEIF localizer is operable **AND** glide slope is operable

THEN

Set runway visibility minimum equal to visibility minimum with both localizer and glide slope operable;

Set runway ceiling minimum equal to ceiling minimum with both localizer and glide slope operable;

IF middle marker is inoperable

THEN

Set runway visibility minimum equal to maximum of runway visibility minimum and runway visibility minimum with localizer and glide slope operable and middle marker inoperable;

Set runway ceiling minimum equal to maximum of runway ceiling minimum and runway ceiling minimum with localizer and glide slope operable and middle marker inoperable;

IF RAIL is inoperable **OR** ALS is inoperable

THEN

Set runway visibility minimum equal to maximum of runway visibility minimum and runway visibility minimum with localizer and glide slope operable and RAIL or ALS inoperable;

Set runway ceiling minimum equal to maximum of runway ceiling minimum and runway ceiling minimum with localizer and glide slope operable and RIAL or ALS inoperable;

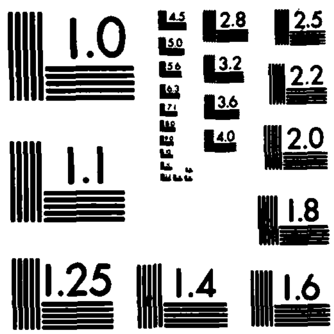
IF TDZ is inoperable

```
                        THEN

                            Set runway visibility minimum equal to maximum of
                            runway visibility minimum and runway visibility minimum
                            with localizer and glide slope operable and TDZ
                            inoperable;

                            Set runway ceiling minimum equal to maximum of runway
                            ceiling minimum and runway ceiling minimum with
                            localizer and glide slope operable and TDZ inoperable;

                    IF  CL is inoperable

                        THEN

                            Set runway visibility minimum equal to maximum of
                            runway visibility minimum and runway visibility minimum
                            with localizer and glide slope operable and CL
                            inoperable;

                            Convert runway ceiling and visibility minima to
                            character form;

        IF  HIRL is inoperable

            THEN

                Set runway visibility minimum equal to 2.0;

                Convert runway visibility minimum to character form;

    ENDREPEAT;

END MINIMA;
```

```
ROUTINE  CLOSING
    [This routine closes runways based on wind conditions and weather minima]

    REPEAT;  (for each runway)

         Close runway based on tower imposed closures;

         Close runway due to wind components exceeding thresholds;

         Close runway due to ceiling and visibility falling below airport minima;

    ENDREPEAT;

END  CLOSING;


ROUTINE  WIND
    [This routine computes crosswind and tailwind components of prevailing wind and sets up corresponding
    screen data fields]

    Convert runway headings from degrees to radians;

    REPEAT;  (for each runway)

         Set runway crosswind component equal to wind velocity multiplied by absolute value of sinus of
         runway heading;

         IF  absolute value of runway heading angle is greater or equal to 1.57079

              THEN  set runway tailwind component equal to zero;

              ELSE      set runway tailwind component equal to wind velocity multiplied by cosine of runway
                        heading;

    ENDREPEAT;

END  WIND;
```

```
ROUTINE  ELIG
   [This routine determines eligibility of configurations based on runway closures, weather conditions, and
   equipment status]

   PERFORM   CONFIGURATION_ID_SET_UP; [setup a number of bit strings signifying dual and triple
                                      configurations]

   Set eligibility bit string to string of zeroes;

   Set number of eligible configurations to zero;

   IF   ceiling is below 100 ft. OR visibility is below .25 mile

        THEN  all configurations are ineligible;

   PERFORM   BELOW_200_CEILING_PLUS_EQUIPMENT_OUTAGE_ELIGIBILITY_CHECK;

   PERFORM   RUNWAY_CLOSURE_ELIGIBILITY_SET_UP;

   REPEAT;   (for each configuration)

        Clear EFLAG;   [set eligibility flag to 'eligible']

        PERFORM   RUNWAY_CLOSURE_ELIGIBILITY_SET_UP;

        IF   configuration is eligible

             THEN

                  PERFORM   BELOW_200_CEILING_ELIGIBILITY_CHECK;

                  IF   configuration is eligible

                       THEN

                            PERFORM   BELOW_1000_CEIL_BELOW_3_VIS_ELIGIBILITY_CHECK;

                            IF   configuration is eligible

                                 THEN
```

AD-A127 398 SOFTWARE DESCRIPTION FOR THE O'HARE RUNWAY 2/3
CONFIGURATION MANAGEMENT SYSTE..(U) MITRE CORP MCLEAN
VA METREK DIV S KAVOUSSI OCT 82 MTR-82W125-VOL-1
UNCLASSIFIED FAA-EM-82-28-VOL-1 DTFA01-81-C-10003 F/G 1/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
PERFORM   BETWEEN_4800_TO_200_CEILING_AND_5_TO_.25_VIS_PLUS_EQUIPMENT_
          OUTAGE_ELGIBILITY_CHECK;

IF   configuration is eligible

     THEN

          PERFORM   HOLD_SHORT_ELIGIBILITY_CHECK;

               Augment eligibility string with EFLAG;
               [construct eligibility bit string]

ENDREPEAT;

Compute number of eligible configurations;

END  ELIG;
```

```
PROCESS  CONFIGURATION_ID_SET_UP
    [This process initializes certain necessary variables for ELIG routine; bit strings are set up to
    indicate parallel, certain duals, triple, and hold short configurations]

END CONFIGURATION_ID_SET_UP;


PROCESS  BELOW_200_CEILING_PLUS_EQUIPMENT_OUTAGE_ELIGIBILITY_CHECK
    [This process determines eligibility of configurations with ceiling below 200 ft and certain equipment
    inoperable]

    IF   prevailing airport ceiling is below 200 ft AND localizer OR outer marker OR middle marker OR ALS on
         runways 14R OR 14L is inoperable

         THEN  all configurations are ineligible;

END BELOW_200_CEILING_PLUS_EQUIPMENT_OUTAGE_ELIGIBILITY_CHECK;


PROCESS  RUNWAY_CLOSURE_ELIGIBILITY_SET_UP
    [This process initializes certain necessary variables for ELIG routine in order to check for
    ineligibility as a result of runway closures and RVR outages]

END  RUNWAY_CLOSURE_ELIGIBILITY_SET_UP;


PROCESS  RUNWAY_CLOSURE_ELIGIBILITY_CHECK
    [This process determines eligibility of configurations due to runway closure]

    IF   one or more of closed runways are in configuration

         THEN  configuration is ineligible;

END  RUNWAY_CLOSURE_ELIGIBILITY_CHECK;
```

```
PROCESS  BELOW_200_CEILING_ELIGIBILITY_CHECK
    [This process determines eligibility of configurations with ceiling below 200]

    IF   ceiling is below 200 ft AND configuration is not parallel 14's

         THEN  configuration is ineligible;

END  BELOW_200_CEILING_ELIGIBILITY_CHECK;


PROCESS  BELOW_.5_VIS_PLUS_NON_RVR_CONFIGURATION_ELIGIBILITY_CHECK
    [This process determines eligibility of configurations with visibility below .5 and non-RVR runways]

    IF   visibility is below .5 AND configuration contains non-RVR runways

         THEN  configuration is ineligible;

END  BELOW_.5_VIS_PLUS_NON_RVR_CONFIGURATION_ELIGIBILITY_CHECK;


PROCESS  BELOW_800_CEIL_2_VIS_ELIGIBILITY_CHECK
    [This process determines eligibility of configurations with ceiling and visibility below 800 and 2
    respectively]

    IF   ceiling is below 800 AND visibility is below 2

         THEN  all non-parallel configurations are ineligible;

END  BELOW_800_CEIL_2_VIS_ELIGIBILITY_CHECK;


PROCESS  BELOW_1000_CEIL_3_VIS_ELIGIBILITY_CHECK;
    [This process determines eligibility of configurations with ceiling and visibility below 1000 and 3
    respectively]

    IF   ceiling is below 1000 AND visibility is below 3

         THEN triple and certain dual (4R, 9R and 9R, 14R) configurations are ineligible;

END  BELOW_1000_CEIL_3_VIS_ELIGIBILITY_CHECK;
```

```
PROCESS  BETWEEN_4800_TO_200_CEILING_AND_5_TO_.25_VISIBILITY_PLUS_EQUIPMENT_OUTAGE_ELIGIBILITY_CHECK;
    [This process determines eligibility of configurations with ceiling between 200 and 4800, visibility
    between .25 and 5 and certain equipment inoperable]

    IF (ceiling is below 1000 AND ceiling is above 200) OR (visibility is below 3 AND visibility is above .25)

        THENIF   any of following is inoperable: glide slope, outer marker, middle marker, or ALS

            THEN  parallel configurations containing runways with above inoperable equipment are ineligible;

    IF (ceiling is below 4800 AND ceiling is above 200) OR (visibility is below 5 AND visibility is above .25)

        THENIF   localizer is inoperable

            THEN  parallel configurations containing runways with localizer inoperable are ineligible;

END  BETWEEN_4800_TO_200_CEILING_AND_5_TO_.25_VISIBILITY_PLUS_EQUIPMENT_OUTAGE_ELIGIBILITY_CHECK;


PROCESS  HOLD_SHORT_ELIGIBILITY_CHECK
    [This process determines eligibility for hold short configurations]

    IF   surface on runway 22R is wet OR braking on runway 22R is poor

         THEN  configuration (22R, 27R arrivals) is ineligible;

    IF   surface on runway 27L is wet OR braking on runway 27L is poor

         THEN  configurations (14R, 27L and 27L, 32L arrivals) are ineligible;

    IF   braking on 14R is poor

         THEN  configurations (14R, 27L and 9R, 14R arrivals) are ineligible;

END  HOLD_SHORT_ELIGIBILITY_CHECK;
```

```
ROUTINE  FILES
    [This routine determines capacity file number for each configuration and sets CNDTN variable to indicate
    VFR (=1) or IFR (=2)]

    REPEAT;   (for each configuration)

         IF   prevailing ceiling is below 800 OR prevailing visibility is below 2

              THEN
                   Set CNDTN to 2;  [indicating IFR conditions]

                   Set capacity file number to 3 indicating IFRDRY file to be used;

                   REPEAT WHILE   (capacity file number is equal to 3);
                        [for each runway]

                        IF   braking is poor

                             THEN  set capacity file number to 4 indicating IFRWET file to be used;

                   ENDREPEAT;

              ELSE
                   Set CNDTN to 1;  [indicating VFR conditions]

                   Set capacity file number to 1 indicating VFRDRY file to be used;

                   REPEAT WHILE  (capacity file number is equal to 1);  [for each runway]

                        IF   braking is poor

                             THEN  set capacity file number to 2 indicating VFRWET file to be used;

                   ENDREPEAT;

    ENDREPEAT;

END  FILES;
```

```
ROUTINE  PERCENT
    [This routine computes north and south demands based on fix-to-runway assignments, also computes
    percentage of arrivals]

    PERFORM   INITIALIZATION (PERCENT);

    Set total arrival demand;

    Set total departure demand;

    REPEAT; (for each configuration)

        Compute total arrival demand for north complex;

        Compute total arrival demand for south complex;

        Compute total departure demand for north complex;

        Compute total departure demand for south complex;

        Compute percentage of arrivals for north complex;

        Compute percentage of arrivals for south complex;

    ENDREPEAT;

END  PERCENT;


PROCESS  INITIALIZAION_(PERCENT)
    [This process performs initialization for PERCENT routine]

END INITIALIZATION_(PERCENT);
```

```
ROUTINE  QFIX
    [This routine updates departure runways for current operating configuration in current departure queue
    screen]

    REPEAT; (for all departure runways in current operating configuration)

        Set new departure runways for use on departure queue screen;

    ENDREPEAT;

END  QFIX;
```

```
ROUTINE  CAPSAT
    [This routine computes capacity and performs demand balancing for each eligible configuration]

    Compute percentage of arrivals for entire airport;

    REPEAT; (for each configuration)

         Clear FLAG;

         IF   configuration is eligible

              THEN

                   PERFORM   CAPACITY_CURVE_SELECTION;  [select proper capacity curve]

                   IF   FLAG is equal to zero

                        THEN

                             CALL  DBAL;  [to balance demand]

                             IF   airport is not saturated

                                  THEN

                                       PERFORM   NORTH_COMPLEX_CAPACITY_CALCULATIONS;
                                                 [using balanced demand]

                                       PERFORM   SOUTH_COMPLEX_CAPACITY_CALCULATIONS;
                                                 [using balanced demand]

                                  ELSE  (saturated)

                                       PERFORM   NORTH_COMPLEX_CAPACITY_CALCULATIONS;
                                                 [using unbalanced demand]

                                       PERFORM   SOUTH_COMPLEX_CAPACITY_CALCULATIONS;
                                                 [using unbalanced demand]

                             IF   FLAG is equal to 1
```

```
                THEN (north only configuration)
                     PERFORM  NORTH_ONLY_CAPACITY_COMPUTATION;
                ELSE  (south only configuration)
                     PERFORM  SOUTH_ONLY_CAPACITY_COMPUTATION;
           PERFORM   CONSTRAIN_CAPACITY_OF_ENTIRE_AIRPORT;
           PERFORM   SATURATION_COMPUTATION;
           PERFORM   CHANGE_DUE_TO_DEMAND_BALANCING_COMPUTATION;
           PERFORM   FINAL_SATURATION_CHECK;
      ELSE  (for ineligible configuration)
           Set special values for capacity;   [used internally to distinguish from eligible
                                               configurations]
  ENDREPEAT;
END  CAPSAT;
```

```
PROCESS  CAPACITY_CURVE_SELECTION
         [This process selects proper capacity curve for north and south complexes]

         Obtain north and south complexes indices;

         REPEAT;   (for each complex)

              Retrieve north and south capacity curves from CAPFILE;

              IF  configuration is north only configuration

                   THEN  set FLAG to 1;

              IF configuration is south only configuration

                   THEN  set flag to 2;

         ENDREPEAT;

END  CAPACITY_CURVE_SELECTION;


PROCESS  NORTH_COMPLEX_CAPACITY_CALCULATIONS
    [This process computes capacity of north complex]

    CALL  CAPCAL;
         [This routine computes arrival and departure capacities of a complex based on percentage of arrivals
         and a particular capacity curve]

END  NORTH_COMPLEX_CAPACITY_CALCULATIONS;


PROCESS  SOUTH_COMPLEX_CAPACITY_CALCULATIONS
    [This process computes capacity of south complex]

    CALL  CAPCAL;
         [This routine computes arrival and departure capacities of a complex based on percentage of arrivals
         and a particular capacity curve]

END  SOUTH_COMPLEX_CAPACITY_CALCULATIONS;
```

```
PROCESS  NORTH_ONLY_CAPACITY_CALCULATION
    [This process computes capacity for north only configurations]

    CALL  CAPCAL;
         [This routine computes arrival and departure capacities of a complex based on percentage of arrivals
         and a particular capacity curve]

         IF   airport is not saturated

              THEN
                   set variables PRCARR and INFORM with their appropriate values derived from demand
                   information and CAPCAL;

END  NORTH_ONLY_CAPACITY_CALCULATION;


PROCESS  SOUTH_ONLY_CAPACITY_CALCULATION
    [This process computes capacity for south only configurations]

    CALL  CAPCAL;
         [this routine computes arrival and departure capacities of a complex based on percentage of arrivals
         and a particular capacity curve]

    IF   airport is not saturated

         THEN
              Set variables PRCARR and INFORM with their appropriate values derived from demand information
              and routine CAPCAL;

END  SOUTH_ONLY_CAPACITY_CALCULATIONS;
```

```
PROCESS  CONSTRAIN_CAPACITY_OF_ENTIRE_AIRPORT
    [This process constrain capacity for entire airport]

    Compute percentage of arrivals for entire airport based on calculated arrival and departure capacities;

    Adjust arrival or departure capacity to conform to percentage of arrivals for entire airport;

END  CONSTRAIN_CAPACITY_OF_ENTIRE_AIRPORT


PROCESS  SATURATION_COMPUTATION
    [This process computes saturation level]

         Compute north complex saturation level;

         Compute south complex saturation level;

         Compute airport saturation level;

END  SATURATION_COMPUTATION;


PROCESS  CHANGE_DUE_TO_DEMAND_COMPUTATION
    [This process computes changes in demand as result of demand balancing]

    IF   airport is not saturated

         THEN
              Compute change in arrival demand between north and south complexes due to demand balancing;

              Compute change in departure demand between north and south complexes due to demand balancing;

END  CHANGE_DUE_TO_DEMAND_BALANCING_COMPUTATION;
```

```
PROCESS  FINAL_SATURATION_CHECK
    [This process checks saturation level and sets appropriate variables]

    IF   airport is saturated

         THEN   set appropriate values for variable PRCARR;

END  FINAL_SATURATION_CHECK;
```

```
ROUTINE   CAPCAL
    [This routine computes arrival and departure capacity of a complex based on percentage of arrivals and a
    particular capacity curve]

    IF   only one capacity point exists OR percentage of arrivals is equal to zero

        THEN  compute departure capacity;

        ELSEIF    percentage of arrivals is less than 100

            THEN  compute arrival and departure capacities;

            ELSE   (all arrivals)

                Compute arrival capacity;

END  CAPCAL;
```

```
ROUTINE  DBAL
    [This routine performs demand balancing]

    REPEAT; [for each airport complex]

        Initialize demand balancing variables;

        CALL  RHO;
             [This routine computes minimum saturation level and balanced arrival and departure demands
             point]

    ENDREPEAT;

    IF  minimum saturation level is greater than 1

        THEN  airport is saturated and balanced demand is not obtained;

        ELSE

             Determine point on either north or south complex capacity curve that corresponds to minimum
             saturation level;

             IF  point of minimum saturation corresponds to whole demand numbers

                  THEN  select those demand numbers as balanced demand figures;

                  ELSE       select closest integer demand numbers that correspond to a saturation level less
                             than 1;

END  DBAL;
```

```
ROUTINE RHO
   [This routine performs demand balancing algorithm]

   REPEAT;   [for each point given on north or south complex capacity curve and each line segment in south
              or north complex capacity curve]

      Compute a coefficient (less than or equal to one) that when multiplied by north and south complex
      curves moves two curves so that point lie on line segment;

   ENDREPEAT;

   Determine smallest such coefficient;     [This coefficient is new balanced saturated point and its
                                             corresponding point is new balanced demand]

   IF  such point does not exist

      THEN  set unbalanced flag;

END RHO;
```

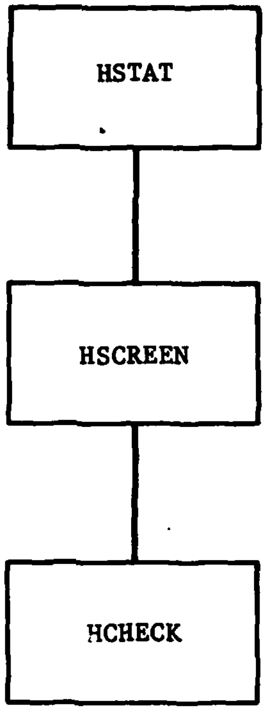


FIGURE A-2
SCHEMATIC DIAGRAM OF
O'HARE STATUS SCREEN ROUTINES

```
ROUTINE  HSTAT
    [This routine prepares information used on O'Hare status summary screen, and stores that information in
    structure OHSTAT]

    Set prevailing airport ceiling;

    Set prevailing airport visibility;

    Set prevailing wind direction;

    Set prevailing wind velocity;

    Set current operating configuration's arrival runways;

    Set current operating configuration's departure runways;

    IF   current operating configuration is ineligible

         THEN  blank out capacity data field on screen and produce appropriate message;

         ELSE  convert capacity from numerical to character data form;

    PERFORM   PERCENTAGE_OF_HIGHEST_CAPACITY_CALCULATION;

    Blank out scroll data field on screen;

    Blank out log messages data fields on screen;

    Initialize structure MESSAGE_MAKER;

    PERFORM   EQUIPMENT_LOG_MESSAGE_GENERATION;

    PERFORM   WEATHER_AND_WIND_LOG_MESSAGE_GENERATION;

    PERFORM   AIRPORT_PLANNING_LOG_MESSAGE_GENERATION;

    PERFORM   LOG_MESSAGE_SORT;

    PERFORM   FLAG_NEW_MESSAGES;
```

```
PERFORM   OLD_MESSAGE_TABLE_GENERATION;

REPEAT UNTIL  (current program status is not equal to PF12);

     Save message in variable ALT1;

     Save scroll function value in variable ALT2;

     REPEAT UNTIL  (current program status is not equal to PF1);

          CALL HSCREEN;
               [This routine controls O'Hare status summary screen]

     ENDREPEAT;

ENDREPEAT;

END  HSTAT;
```

```
PROCESS  PERCENTAGE_OF_HIGHEST_CAPACITY_CALCULATION
    [This process computes percentage of capacity of current operating configuration to highest available
    capacity]

    Determine maximum of available capacity;

    Compute percentage of high capacity by dividing current operating configuration's capacity to maximum
    available capacity;

END  PERCENTAGE_OF_HIGHEST_CAPACITY_CALCULATION;


PROCESS  EQUIPMENT_LOG_MESSAGE_GENERATION
    [This process generates appropriate log messages for O'Hare status summary screen from equipment
    planning log information]

    REPEAT;   [up to 15 possible equipment planning log messages]

        IF   a message is found

             THEN
                  Construct a new message for O'Hare status summary screen using runway ID, equipment type,
                  out of service time, return to service time, and remarks;

                  Increment message counter;

    ENDREPEAT;

END  EQUIPMENT_LOG_MESSAGE_GENERATION;
```

```
PROCESS  WEATHER_AND_WIND_LOG_MESSAGE_GENERATION
    [This process generates appropriate log messages for O'Hare status summary screen from weather and wind
    planning log information]

    REPEAT;    [up to 13 possible weather and wind planning log messages]

        IF    a message is found

            THEN
                Construct new messages for O'Hare status summary screen using ceiling and/or visibility
                minima and wind direction and/or velocity, time, and remarks;

                Increment message counter;

    ENDREPEAT;

END  WEATHER_AND_WIND_LOG_MESSAGE_GENERATION;


PROCESS  AIRPORT_PLANNING_LOG_MESSAGE_GENERATION
    [This process generates appropriate log messages for O'Hare status summary screen from airport planning
    log information]

    REPEAT;    [up to 13 possible airport planning log messages]

        IF    a message is found

            THEN
                Construct new messages for O'Hare status summary screen using time, runway ID,
                braking/surface conditions, runway closures information, and remarks;

                Increment message counter;

    ENDREPEAT;

END  AIRPORT_PLANNING_LOG_MESSAGE_GENERATION;
```

```
PROCESS  LOG_MESSAGE_SORT
    [This process sorts O'Hare status summary screen messages based on time]

END  LOG_MESSAGE_SORT;


PROCESS  FLAG_NEW_MESSAGES
    [This process determines which messages are newly added or modified in order to highlight them on O'Hare
    status summary screen]

    Compare contents of MESSAGE_MAKER with OLDMES;

    IF   a new message is found

         THEN  set flag INKEEP equal to one for that message;

END  FLAG_NEW_MESSAGES;


PROCESS  OLD_MESSAGE_TABLE_GENERATION
    [This process copies contents of MESSAGE_MAKER into OLDMES constructing a copy of current messages to be
    used on next cycle as old message table]

END  OLD_MESSAGE_TABLE_GENERATION;
```

```
ROUTINE  HSCREEN
   [This routine controls O'Hare status summary screen]

   Set data masks to normal intensity;

   Set message data mask to high intensity;

   Set cursor equal to value 14; (on scroll data field)

   PERFORM   SET_UP_SCREEN_POINTERS_(HSTAT);

   Determine current time;

   PERFORM   CHARACTER_TO_NUMERICAL_CONVERSION_OF_TIME;

   PERFORM   DIVISION_OF_LOG_MESSAGES_BASED_ON _CURRENT_TIME;

   REPEAT UNTIL   (current program status is not equal to ENTER);

       IF   There are log messages available

           THEN

               PERFORM   SCROLL_FUNCTION_SET_UP;

               PERFORM

           ELSE  (if no log messages exist)

               PERFORM   SET_UP_NULL_SCREEN_MESSAGES;

           PERFORM   DISPLAY_PANEL;

           IF   current program status is equal to ENTER

               THEN

                   Set data masks to normal intensity;

                   Set message data mask to high intensity;
```

```
CALL  HCHECK;
      [This routine checks for errors occurred on screen as a result of an erroneous
      entry and returns an appropriate screen message advising user with corrections]

IF    screen message is not equal to 'DATA ENTERED'

      THEN  highlight erroneous entry;

      ELSE  add current time to screen message;

   ENDREPEAT;

END  HSCREEN;
```

```
PROCESS  SET_UP_SCREEN_POINTERS_(HSTAT)
    [This process sets up screen pointers for DMS use]

    Structure OH_LOADLIST is set up with address of location of OHSTAT variables;  [DMS uses OH_LOADLIST to
    load and unload data onto and from screen]

END  SET_UP_SCREEN_POINTERS_(HSTAT);


PROCESS  CHARACTER_TO_NUMERICAL_CONVERSION_OF_TIME
    [This process converts time from character to numerical data]

END  CHARACTER_TO_NUMERICAL_CONVERSION_OF_TIME;


PROCESS  DIVISION_OF_LOG_MESSAGES_BASED_ON_CURRENT_TIME
    [This routine divides log messages into two segments:  past and future based on current running time]

    Compute number of messages with associated times less than current running time;

    Compute number of messages with associated times more than current running time;

END  DIVISION_OF_LOG_MESSAGES_BASED_ON_CURRENT_TIME;
```

```
PROCESS  SCROLL_FUNCTION_SET_UP
    [This process performs scrolling function associated with O'Hare status summary screen by setting up
    pointers to first and last messages to appear on screen at one time]

    Add value of scroll data field to pointers signifying first and last messages to appear on screen;
    (negative scroll numbers are allowed)

    Perform bookkeeping to determine how many messages are to be displayed and their locations on list of
    messages;

END  SCROLL_FUNCTION_SET_UP;


PROCESS  SET_UP_SCREEN_LOG_MESSAGES
    [This process determines screen messages to be displayed based on pointers calculated before, and sets
    up pointers for DMS use; and constructs message headings]

END  SET_UP_SCREEN_LOG_MESSAGES;


PROCESS  SET_UP_NULL_SCREEN_MESSAGE
    [This process issues a message indicating there are no log messages]

END  SET_UP_NULL_SCREEN_MESSAGE;


PROCESS  DISPLAY_PANEL
    [This process invokes PDISPLAY preprocessor which in turn interfaces with DMS panel manager and displays
    requested screen]

END  DISPLAY_PANEL;
```

```
ROUTINE  HCHECK
    {This routine checks for errors occurred on screen as a result of an erroneous entry and returns an
    appropriate screen message advising user with corrections}

    IF  character data is detected in scroll data field

        THEN  issue message 'NUMERIC INPUT REQUIRED';

    IF  a decimal point is detected in scroll data field

        THEN  issue message 'NO DECIMAL POINTS ALLOWED';

END  HCHECK;
```

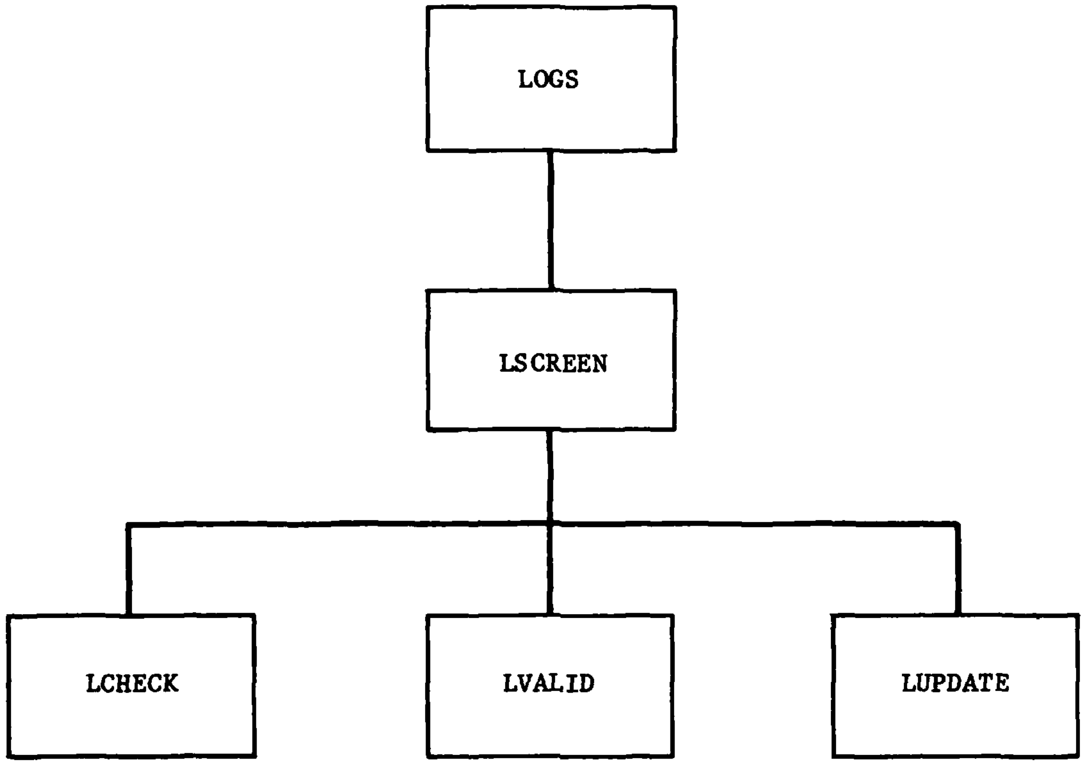


FIGURE A-3
SCHEMATIC DIAGRAM OF
PLANNING LOG SELECTION SCREENS

```
ROUTINE  LOGS
    [This routine invokes planning log selection screen]

    Initialize screen data fields to blanks;

    REPEAT UNTIL   (current program status is not equal to PF12);

        Save a copy of LOGNUM structure for 'restore previous screen' function;

        REPEAT UNTIL   (current program status is not equal to PF2);

            CALL LSCREEN;
                [This routine controls planning log selection screens]

        ENDREPEAT;

    ENDREPEAT;

END LOGS;


ROUTINE  LSCREEN
    [This routine controls planning log selection screen]

    PERFORM   SET_UP_SCREEN_POINTERS_(LOGS);

    Set data makes to normal intensity;
    Set message data mask to high intensity;
    Set cursor equal to value 1; (1st data field on screen)

    REPEAT UNTIL   (current program status is not equal to ENTER);

        PERFORM   DISPLAY_PANEL;

        IF   current program status is equal to PA1

            THEN stop;
```

```
IF   current program status is equal to ENTER

     THEN

          Set data masks to normal intensity;
          Set message data mask to high intensity;

          CALL LCHECK;
               [This routine checks for errors occurred on screen as a result of an erroneous entry
               and returns value for cursor pointing to first data field where an error has
               occurred; and an appropriate screen message is issued advising user with corrections]

          IF   screen message is not equal to 'DATA ENTERED'

               THEN highlight erroneous entry;

               ELSE

                    CALL LVALID;
                         [This routine performs data validation checks on screen entries and
                         returns value for cursor pointing to first invalid data field.  Also, an
                         appropriate screen message is issued advising user with corrections]

                    IF   screen message is not equal to 'DATA ENTERED'

                         THEN highlight inconsistent entry;

                         ELSE

                              CALL LUPDATE;
                                   [This routine is performed only when there are no errors
                                   committed on screen, it updates appropriate variables pertaining
                                   to this screen program with new screen entries.  Also, returns
                                   new value of current program status]

     ENDREPEAT;

END LSCREEN;
```

PROCESS SET_UP_SCREEN_POINTERS_(LOGS)
[This process sets up screen pointers for DMS use]

Structure LOG_LOADLIST is set up with address of location of LOGNUM_DATA variables; (DMS uses LOG LOADLIST to load and unload data onto and from screen)

END SET_UP_SCREEN_POINTERS_(LOGS);

```
ROUTINE  LCHECK
    [This routine checks for errors occurred on screen as a result of an erroneous entry and returns value
    for cursor pointing to first data field where an error has occurred; and an appropriate screen message
    is issued advising user with corrections]

    REPEAT WHILE   (screen message is equal to 'DATA ENTERED');
                   [for each data field on screen]

        Increment cursor;

        IF   an alphanumeric data other than 'X' or blank is detected

            THEN issue message 'INPUT MUST BE X OR BLANK';

    ENDREPEAT;

END LCHECK;
```

ROUTINE LVALID
[This routine performs data validation checks on screen entries and returns value for cursor pointing to first invalid data field. Also, an appropriate screen message is issued advising user with corrections]

IF more than one 'X' is detected on screen

THEN issue message 'SELECT ONLY ONE PLANNING LOG';

END LVALID;

ROUTINE LUPDATE
[This routine is performed only when there are no errors committed on screen, it updates appropriate variables pertaining to this screen program with new screen entries, also, returns new value of RSTATUS]

IF weather and wind planning log is selected

THEN set current program status to PF13;

IF equipment planning log is selected

THEN set current program status to PF15;

IF airport planning log is selected

THEN Set current program status to PF14;

IF demand planning log is selected

THEN set current program status to PF16;

END LUPDATE;

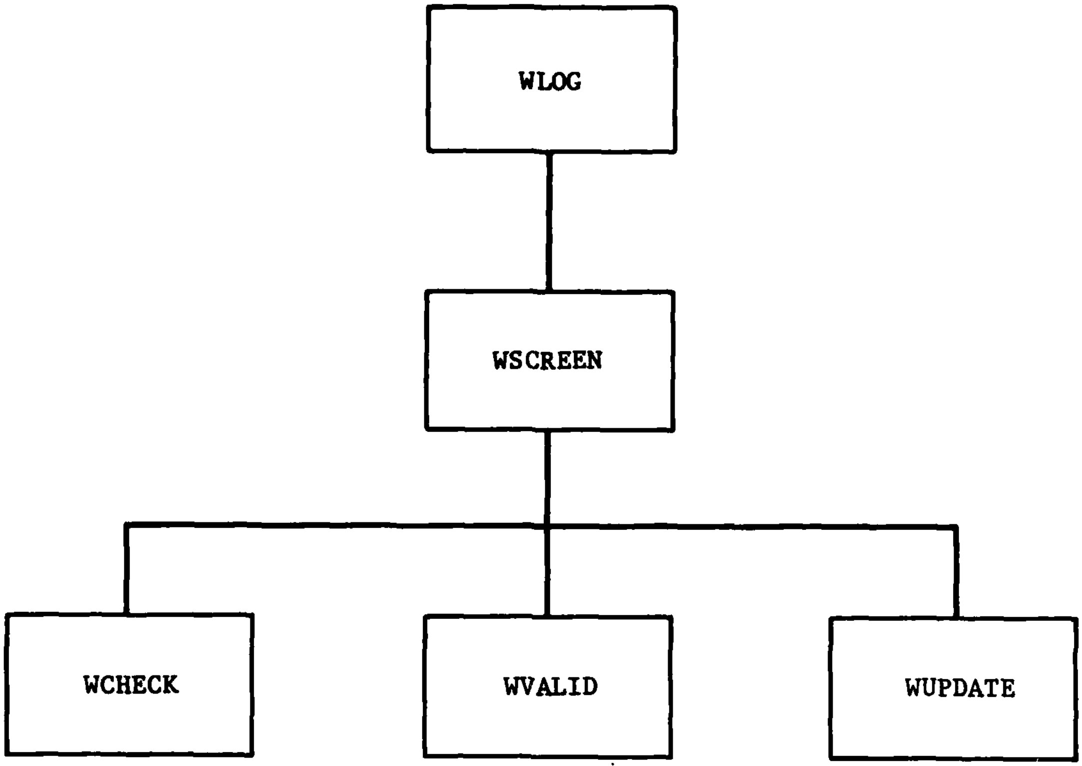


FIGURE A-4
SCHEMATIC DIAGRAM OF
WEATHER AND WIND PLANNING LOG SCREEN ROUTINES

```
ROUTINE  WLOG
    [This routine invokes weather and wind planning log screen]

    REPEAT UNTIL  (current program status is not equal to PF12);

        Save a copy of WXLOG and CNVTWX structures for 'restore previous screen' function;

        CALL  WSCREEN;
            [This routine controls weather and wind planning log screen]

    ENDREPEAT;

    IF  Screen message is equal to 'DATA ENTERED'

        THEN  update WXLOG and CNVTWX structures with new screen values;

END  WLOG;
```

```
ROUTINE  WSCREEN
    [This routine controls weather and wind planning log screen]

    Set data masks to default intensity (normal);
    Set message data mask to high intensity;
    Set cursor to position 61; (1st data field used)

    PERFORM   SET_UP_SCREEN_POINTERS_(WLOG);

    REPEAT UNTIL  (current program status is not equal to ENTER);

         PERFORM   DISPLAY_PANEL;

         IF   current program status is equal to PA1

              THEN  stop;

         IF   current program status is equal to ENTER

              THEN

                   Set data masks to normal intensity;
                   Set message data mask equal to high intensity;

                   CALL  WCHECK;
                        [This routine checks for errors occured on screen as a result of an
                        erroneous entry and returns value for cursor pointing to first data field
                        where an error has occured; and an appropriate screen message is issued
                        advising user with corrections]

                   IF   screen message is not equal to 'DATA ENTERED'

                        THEN  highlight erroneous entry;

                        ELSE

                             CALL  WVALID;
                                  [This routine performs data validation checks on screen entries
                                  and returns value for cursor pointing to first invalid data
                                  field.  Also, an appropriate screen message is issued advising
                                  user with corrections]
```

```
            IF   screen message is not equal to 'DATA ENTERED'

                 THEN  highlight inconsistant entry;

                 ELSE

                      CALL  WUPDATE;
                            [This routine is performed only when there are no
                            errors committed on screen, it sorts log screen
                            entries based on time]

                      Add current time to screen message;

    ENDREPEAT;

END  WSCREEN;


PROCESS  SET_UP_SCREEN_POINTERS_(WLOG)
    [This process sets up screen pointers for DMS use]

    Structure WX_LOADLIST is set up with address of location of WX_DATA variables;  [DMS uses WX
    LOADLIST to load and unload data onto and from screen]

END  SET_UP_SCREEN_POINTERS_(WLOG);
```

```
ROUTINE  WCHECK
    [This routine checks for errors occured on screen as a result of an erronesoun entry and
    returns value for cursor pointing to first data field where an error has occured; and an
    appropriate screen message is issued advising user with corrections]

    IF   a character data is detected in any of numeric data fields

         THEN  issue message 'NUMERIC INPUT REQUIRED';

    REPEAT WHILE   (screen message is equal to 'DATA ENTERED');
         [for each log message designated to this user]

         Increment cursor;

         PERFORM   TIME_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   CEIL_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   VIS_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   DIR_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;
```

```
        PERFORM   VEL_DATA_FIELD_ERROR_CHECK;
        EXITIF    error is detected
        Increment cursor;
    ENDREPEAT;
END  WCHECK;
```

```
PROCESS  TIME_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on time data field; if an error has occured an appropriate
    message is issued]

END  TIME_DATA_FIELD_ERROR_CHECK;


PROCESS  CEIL_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on ceiling data field; if an error has occured an appropriate
    message is issued]

END  CEIL_DATA_FIELD_ERROR_CHECK;


PROCESS  VIS_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on visibility data field; if an error has occured an
    appropriate message is issued]

END  VIS_DATA_FIELD_ERROR_CHECK


PROCESS  DIR_DATA_FIELD_ERROR_CHECK;
    [This process checks for errors on wind direction data field; if an error has occured an
    appropriate message is issued]

END  DIR_DATA_FIELD_ERROR_CHECK;


PROCESS  VEL_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on wind velocity data field; if an error has occured an
    appropriate message is issued]

END  VEL_DATA_FIELD_ERROR_CHECK;
```

```
ROUTINE  WVALID
   [This routine performs data validation check on screen entries and returns value for cursor
   pointing to invalid data field.  Also, an appropriate screen message is issued advising user
   with corrections]

   REPEAT WHILE  (screen message is equal to 'DATA ENTERED');
        [for each log message designated to this user]

        IF   all entries are blank

             THEN

                  move cursor to next line of messages;

                  Set numerical value of time associated with this message equal to an arbitrary
                  large number;

             ELSE

                  Increment cursor;

                  IF   all entries other than time are blank

                       THEN  issue message 'ADDITIONAL INFORMATION REQUIRED FOR THIS TIME ENTRY';

                       ELSEIF    time entry is blank

                            THEN  issue message 'SPECIFY TIME ASSOCIATED WITH ENTRIES';

                            ELSE

                                 PERFORM   TIME_CHECK;

                                 EXITIF    error is detected

                                 Increment cursor;

                                 PERFORM   LEFT_ZERO_PADDING_ON_TIME_ENTRY;

                                 IF   screen message is equal to 'DATA ENTERED'
```

```
THEN
     Increment cursor;

     IF   ceiling entry is not blank

          THEN

               PERFORM   RIGHT_JUSTIFY_CEIL_DATA_ENTRY;

               Increment cursor;

     IF   visibility entry is not blank

          THEN

               PERFORM   RIGHT_JUSTIFY_VIS_DATA_ENTRY;

               Increment cursor;

     IF   wind direction entry is not blank

          THENIF  Wind direction exceeds 360

               THEN      issue message 'WIND DIRECTION MUST
                         NOT EXCEED 360 DEGREES';

     IF   screen message is not equal 'DATA ENTERED'

          THEN
               Increment cursor;

               IF   wind velocity entry is not blank

                    THEN PERFORM   RIGHT_JUSTIFY_VEL_DATA
                                   ENTRY;

               Increment cursor;

               PERFORM   LEFT_JUSTIFY_REMARKS_DATA_ENTRY;

ENDREPEAT;

END  WVALID;
```

```
PROCESS  TIME_CHECK
    [This process checks validity of time entry; if invalid, issues an appropriate message]

END  TIME_CHECK;


PROCESS  LEFT_ZERO_PADDING_ON_TIME_ENTRY
    [This process pads time entry with leading zeroes]

END  LEFT_ZERO_PADDING_ON_TIME ENTRY;


PROCESS  RIGHT_JUSTIFY_CEIL_DATA_ENTRY
    [This process right-justifies ceiling entry]

END  RIGHT_JUSTIFY CEIL_DATA_ENTRY;


PROCESS  LEFT_ZERO_PADDING_ON_WIND_DIRECTION_ENTRY
    [This process pads wind direction entry with leading zeroes]

END  LEFT_ZERO_PADDING_ON_WIND_DIRECTION_ENTRY


PROCESS  LEFT_JUSTIFY_REMARKS_DATA_ENTRY
    [This process left-justifies remarks field]

END  LEFT_JUSTIFY_REMARKS_DATA_ENTRY;


PROCESS  RIGHT_JUSTIFY_VEL_DATA_ENTRY
    [This process right-justifies wind velocity entry]

END  RIGHT_JUSTIFY_VEL_DATA_ENTRY;
```

__ROUTINE__ WUPDATE
    [This routine is performed only when there are no errors committed on screen, it sorts screen
    log entries based on time]

__END__ WUPDATE;

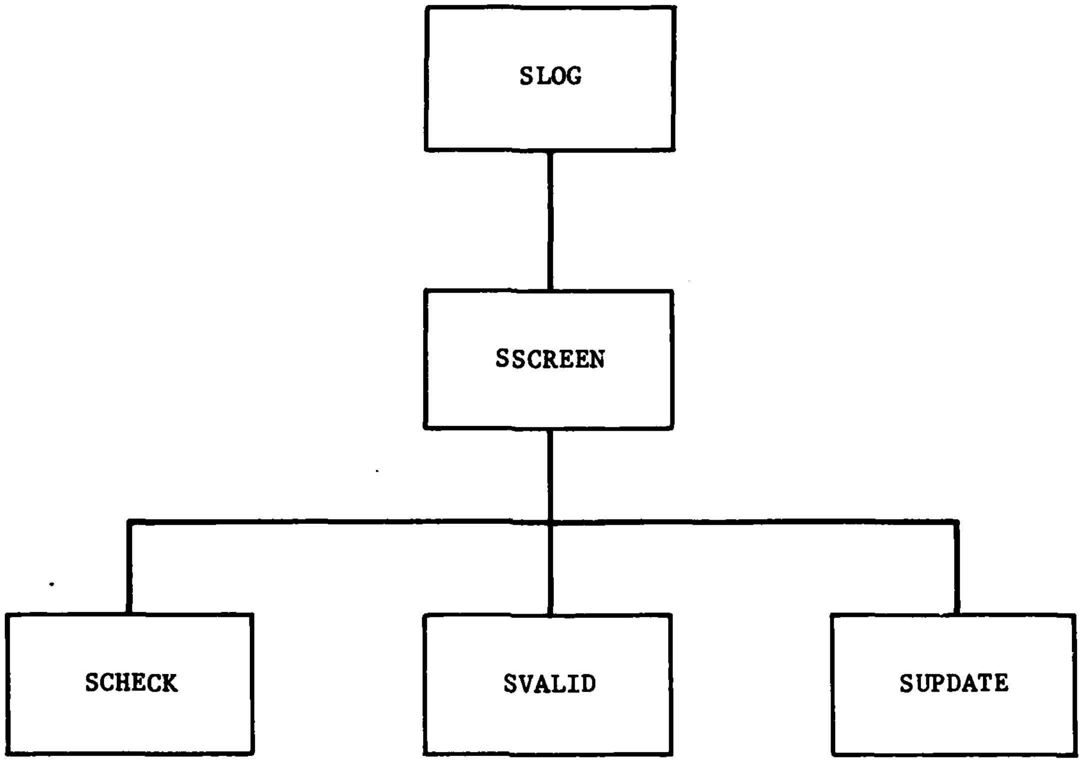


FIGURE A-5
SCHEMATIC DIAGRAM OF
SURFACE CONDITIONS PLANNING LOG SCREEN ROUTINES

```
ROUTINE      SLOG
    [This routine invokes airport surface and runway planning log screen]

    REPEAT UNTIL   (current program status is not equal to PF12);

        Save a copy of SURFLOG and CNVTSRF structures for 'restore previous screen' function;

        CALL  SSCREEN;
             [This routine controls airport surface and runway planning log screen]

    ENDREPEAT;

    IF   screen message is equal to 'DATA ENTERED'

        THEN  update SURFLOG and CNVTSRF structures with new screen values;

END  SLOG;
```

```
ROUTINE     SSCREEN
    [This routine controls airport surface and runway planning log screen]

    Set data masks to normal intensity;

    Set message data mask to high intensity;

    Set cursor to position 71;    (first data field used by user)

    PERFORM   SET_UP_SCREEN_POINTERS_(SLOG);

    REPEAT UNTIL   (current program status is nt equal to ENTER);

        PERFORM   DISPLAY_PANEL;

        IF   current program status is equal to PA1

             THEN  stop;

        IF   current program status is equal to ENTER

             THEN

                  Set data masks equal to normal intensity;

                  Set message data mask equal to high intensity;

                  CALL  SCHECK;
                       [This routine checks for errors occured on screen as a result of an
                       erroneous entry and returns value for cursor pointing to first data field
                       where an error has occured, and an appropriate screen message is issued
                       advising user with corrections]

                  IF   screen message is not equal to 'DATA ENTERED'

                       THEN  highlight erroneous entry;

                       ELSE

                            CALL  SVALID;
                                 [This routine performs data validation checks on screen entries
                                 and returns value for cursor pointing to first invalid data
                                 field.  Also, an appropriate screen message is issued advising
                                 user with corrections]
```

```
IF  screen message is not equal to 'DATA ENTERED'

    THEN  highlight inconsistant entry;

    ELSE

        CALL  SUPDATE;
              [This routine is performed only when there are no
              errors committed on screen, it sorts screen log
              entries based on time]

        Add current time to screen message;

ENDREPEAT;

END  SSCREEN;
```

```
PROCESS  SET_UP_SCREEN_POINTERS_(SLOG)
    [This process sets up screen pointers for DMS use]

    Structure SURF_LOADLIST is set up with address of location of SURF_DATA variables;  [DMS uses
    SURF_LOADLIST to load and unload data onto and from screen]

END      SET_UP_SCREEN_POINTERS_(SLOG);
```

```
ROUTINE  SCHECK
    [This routine checks for errors occured on screen as a result of an erroneous entry and
    returns value for cursor pointing to first data field where an error has occured; and an
    appropriate screen message is issued advising user with corrections]

    IF   a character data is detected in any of numeric data fields

         THEN  issue message 'NUMERIC INPUT REQUIRED';

    REPEAT WHILE   (screen message is equal to 'DATA ENTERED');
         [for each log message designated to this user]

         Increment cursor;

         PERFORM   TIME_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   RUNWAY_ID_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

    ENDREPEAT;

END  SCHECK;
```

```
PROCESS  RUNWAY_ID_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on runway ID data field; if an error has occured, an
    appropriate message is issued]

    END  RUNWAY_ID_DATA_FIELD_ERROR_CHECK;


PROCESS  TIME_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on time data field; if an error has occured, an appropriate
    message is issued]

END  TIME_DATA_FIELD_ERROR_CHECK;
```

```
ROUTINE  SVALID
    [This routine performs data validation checks on screen entries and returns value for cursor
    pointing to first invalid data field.  Also, an appropriate screen message is issued advising
    user with corrections]

    REPEAT WHILE  (screen message is equal to 'DATA ENTERED');
         [for each log message designated to this user]

    IF   all entries are blank

         THEN

              Move cursor to next line of messages;

              Set numerical value of time associated with this message equal to an arbitrary large
              number;

         ELSE

              Increment cursor;

              IF   all entries other than time and runway ID are blank

                   THEN  issue message 'ADDITIONAL INFORMATION REQUIRED FOR THIS ENTRY';

                   ELSE

                        Increment cursor;
```

```
IF   time entry is blank
     THEN  issue message 'SPECIFY TIME ASSOCIATED WITH ENTRIES';
     ELSE
          Increment cursor;
          IF   runway ID entry is blank
               THEN  issue message 'SPECIFY RUNWAY ASSOCIATED WITH
               ENTRIES';
               ELSE
                    Decrement cursor;
                    PERFORM   TIME_CHECK;
                    EXITIF    time entry is invalid
                    PERFORM   LEFT_ZERO_PADDING_ON_TIME_ENTRY;
                    IF   screen message is not equal to 'DATA ENTERED'
                         THEN
                              PERFORM  RIGHT_JUSTIFY_RWY_DATA_ENTRY;
                              PERFORM  LEFT_JUSTIFY_SURF_DATA_ENTRY;
                              PERFORM  LEFT_JUSTIFY_BRAK_DATA_ENTRY;
                              PERFORM  LEFT_JUSTIFY_CLOSED_DATA_ENTRY;
                              PERFORM  LEFT_JUSTIFY_OPEN_DATA_ENTRY;
                              PERFORM  LEFT_JUSTIFY_REMARKS_DATA_ENTRY;
  ENDREPEAT;
END  SVALID;
```

```
PROCESS  TIME_CHECK
    [This process checks validity of time entry; if time entry is found invalid, an appropriate
    message is issued]

END;  TIME_CHECK;


PROCESS  LEFT_ZERO_PADDING_ON_TIME_ENTRY
    [This process pads time entry with leading zeros]

END  LEFT_ZERO_PADDING_ON_TIME_ENTRY;


PROCESS  RIGHT_JUSTIFY_RWY_DATA_ENTRY
    [This process right-justifies runway ID entry]

END  RIGHT_JUSTIFY_RWY_DATA_ENTRY;


PROCESS  LEFT_JUSTIFY_SURF_DATA_ENTRY
    [This process left-justifies surface conditions entry]

END  LEFT_JUSTIFY_SURF_DATA_ENTRY;


PROCESS  LEFT_JUSTIFY_BRAK_DATA_ENTRY
    [This process left-justifies braking conditions entry]

END  LEFT_JUSTIFY_BRAK_DATA_ENTRY;
```

```
PROCESS  LEFT_JUSTIFY_CLOSED_DATA_ENTRY
    [This process left-justifies runway closure entry]

END  LEFT_JUSTIFY_CLOSED_DATA_ENTRY;


PROCESS  LEFT_JUSTIFY_OPEN_DATA_ENTRY
    [This process left-justifies runway opening entry]

END  LEFT_JUSTIFY_OPEN_DATA_ENTRY;


PROCESS  LEFT_JUSTIFY_REMARKS_DATA_ENTRY
    [this process left-justifies remarks entry]

END  LEFT_JUSTIFIES_REMARKS_DATA_ENTRY;
```

```
ROUTINE  SUPDATE
    [This routine is performed only when there are no errors committed on screen, it sorts screen
    log entries based on time]

END  SUPDATE;
```

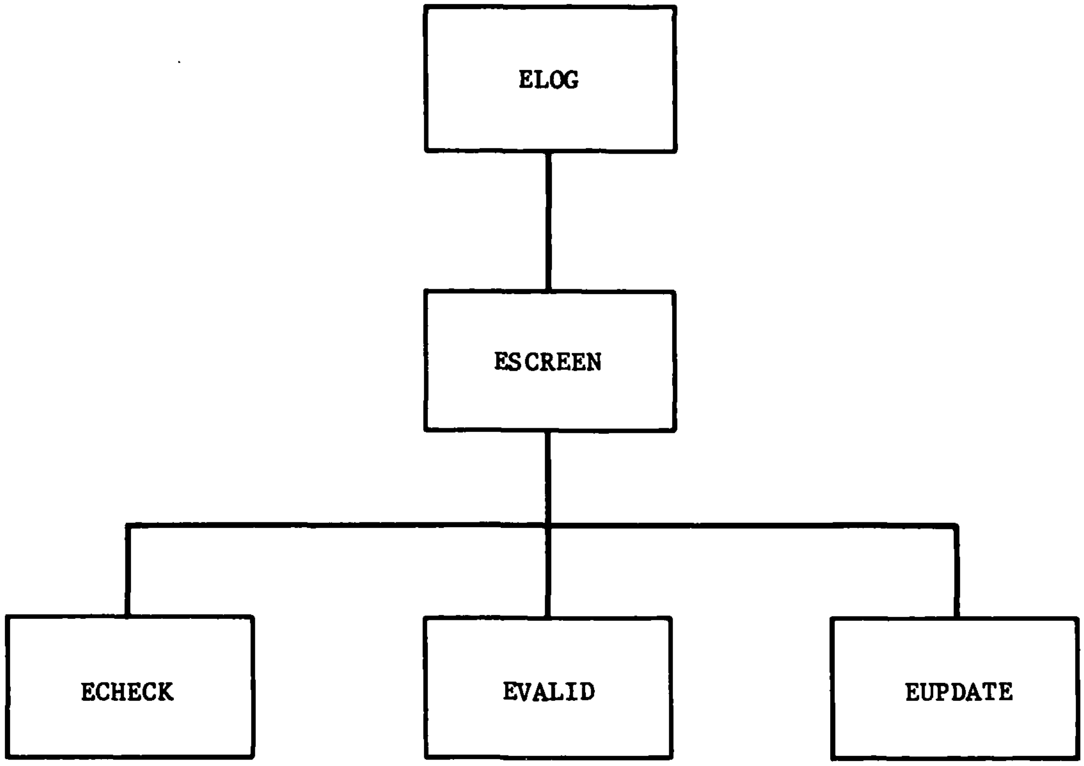


FIGURE A-6
SCHEMATIC DIAGRAM OF
EQUIPMENT PLANNING LOG ROUTINES

```
ROUTINE  ELOG
    [This routine invokes equipment planning log screen]

    REPEAT UNTIL   (current program status is not equal to PF12);
        Save a copy of EQPLOG and CNVTEQP structures for 'restore previous screen' function;

        CALL ESCREEN;
            [This routine controls equipment planning log screen]

    ENDREPEAT;

    IF   screen message is equal to 'DATA ENTERED'

        THEN  Update EQPLOG and CNVTEQP structures with new screen values;

END  ELOG;
```

```
ROUTINE  ESCREEN
    [This routine controls equipment planning log screen]

    Set data masks to normal intensity;

    Set message date mask to high intensity;

    Set cursor to position 61; (1st data field used)

    PERFORM  SET_UP_SCREEN_POINTERS_(ELOG);

    REPEAT UNTIL  (current program status is not equal to ENTER);

        PERFORM  DISPLAY_PANEL;

        IF  current program status is equal to PA1

            THEN  stop;

        IF  current program status is equal to ENTER

            THEN

                Set data masks to normal intensity;

                Set message data mask to high intensity;

                CALL  ECHECK;
                    [This routine checks for errors occured on screen as a result of an
                    erroneous entry and returns value for cursor pointing to first data field
                    where an error has occured; and an appropriate screen message is issued
                    advising user with corrections]

                IF  screen message is not equal to 'DATA ENTERED'

                    THEN  highlight erroneous entry;

                    ELSE
```

```
CALL  EVALID;
      [This routine performs data validation checks on screen entries
      and returns value for cursor pointing to first invalid data
      field.  Also, an appropriate screen message is issued advising
      user with corrections]

IF    screen message is not equal to 'DATA ENTERED'

      THEN  highlight inconsistant entry:

      ELSE

            CALL  EUPDATE;
                  [This routine is performed only when there are no
                  errors committed on screen, it sorts screen log
                  entries based on primarily OTS and then RTS times]

            Add current time to screen message;

   ENDREPEAT;

END  ESCREEN;
```

```
PROCESS  SET_UP_SCREEN_POINTERS_(ELOG)
    [This process sets up screen pointers for DMS use]

    Structure EQUIP_LOADLIST is set up with address of location of EQUIP_DATA variables;
    [DMS uses EQUIP_LOADLIST to load and unload data onto and from screen]

END  SET_UP_SCREEN_POINTERS_(ELOG);
```

```
ROUTINE  ECHECK
    [This routine checks for errors occured on screen as a result of an erroneous entry and
    returns value for cursor pointing to first data field when an error has occured; and an
    appropriate screen message is issued advising user with corrections]

    REPEAT WHILE   (sceen message is equal to 'DATA ENTERED');
         [for each log message designated to this user]

         Increment cursor;

         PERFORM   RUNWAY_ID_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   OTS_TIME_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM  RTS_TIME_DATA_FIELD_ERROR_CHECK;

    ENDREPEAT;

END  ECHECK;
```

```
PROCESS RUNWAY_ID_DATA_FIELD_ERROR_CHECK;
    [This process checks for errors on runway ID data field; if an error has occured, an
    appropriate message is issued]

END RUNWAY_ID_DATA_FIELD_ERROR_CHECK;


PROCESS OTS_TIME_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on OTS time data field; if an error has occured, an
    appropriate message is issued]

END OTS_TIME_DATA_FIELD_ERROR_CHECK;


PROCESS RTS_TIME_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on RTS time data field; if an error has occured, an
    appropriate message is issued]

END RTS_TIME_DATA_FIELD_ERROR_CHECK;
```

```
ROUTINE EVALID
   [This routine performs data validation checks on screen entries and returns value for cursor
   pointing to first invalid data field.  Also, an appropriate screen message is issued advising
   user with corrections]

   REPEAT WHILE  (screen message is equal to 'DATA ENTERED');
        [for each log message designated to this user]

        IF  all entries are blank

            THEN

                 Move cursor to next line of messages;

                 Set numerical value of OTS and RTS times associated with this message equal to
                 arbitrary large numbers; [for purpose of sorting messages later]

            ELSE

                 Increment cursor;

                 PERFORM  RIGHT_JUSTIFY_RWY_DATA_ENTRY;

                 IF  all entries other than runway ID are blank

                     THEN  issue message 'ADDITIONAL INFORMATION REQUIRED FOR THIS RUNWAY';

                     ELSEIF  runway ID entry is blank

                          THEN  issue message 'SPECIFY RUNWAY ASSOCIATED WITH ENTRIES';

                          ELSEIF   equipment type entry is blank

                               THEN  issue message 'SPECIFY EQUIPMENT TYPE ASSOCIATED WITH
                               ENTRIES';

                               ELSEIF   OTS and/or RTS time entries is blank

                                    THEN  issue message "AN OTS AND/OR RTS TIME IS REQUIRED';

                                    ELSE

                                         Increment cursor;
```

```
PERFORM RIGHT_JUSTIFY_EQUIPMENT_DATA_ENTRY;

IF OTS time entry is blank

    THEN set numerical value of OTS time to an arbitrary large
         number;

    ELSE

        PERFORM   OTS_TIME_CHECK;

        EXITIF    error is detected

        PERFORM   LEFT_ZERO_PADDING_ON_OTS_TIME_ENTRY;

        IF   screen message is equal to 'DATA ENTERED'

             THEN

                 Increment cursor;

                 IF   RTS time entry is blank

                      THEN set numerical value of RTS time to
                           an arbitrary large number;

                      ELSE

                          PERFORM   RTS_TIME_CHECK;

                          EXITIF    errors is detected

                          PERFORM   LEFT_ZERO_PADDING_ON_RTS
                                    TIME_ENTRY;

                          IF   screen message is equal to
                               blanks
```

```
THENIF     both RTS and OTS
           time entries are
           not blank AND
           numerical value of
           OTS time is greater
           than numerical
           value of RTS time

      THEN issue message 'TIME
           FOR RTS MUST BE
           BLANK OR LATER THAN
           OTS';

      ELSE

           Increment cursor;

           PERFORM
                LEFT_JUSTIFY
                REMARKS_DATA
                ENTRY;

    ENDREPEAT;

END EVALID;
```

```
PROCESS  RIGHT_JUSTIFY_RWY_DATA_ENTRY
    [This process right-justifies runway ID entry]

END  RIGHT_JUSTIFY_RWY_DATA_ENTRY;


PROCESS  RIGHT_JUSTIFY_EQUIPMENT_DATA_ENTRY
    [This process right-justifies equipment type entry]

END  RIGHT_JUSTIFY_EQUIPMENT_DATA_ENTRY;


PROCESS  OTS_TIME_CHECK
    [This process checks for validity of OTS time entry; if entry is invalid, an appropriate
    message is issued]

END  OTS_TIME_CHECK;


PROCESS  LEFT_ZERO_PADDING_ON_OTS_TIME_ENTRY
    [This process pads OTS time entry with leading zeroes]

END  LEFT_ZERO_PADDING_ON_OTS_TIME_ENTRY;


PROCESS  RTS_TIME_CHECK
    [This process checks for validity of RTS time entry; if entry is invalid, an appropriate
    message is issued]

END  RTS_TIME_CHECK;


PROCESS  LEFT_ZERO_PADDING_ON_RTS_TIME_ENTRY
    [This process pads RTS time entry with leading zeroes]

END  LEFT_ZERO_PADDING_ON_RTS_TIME_ENTRY;
```

```
PROCESS  LEFT_JUSTIFY_REMARKS_DATA_ENTRY
    [This process left justifies remarks entry]

END  LEFT_JUSTIFY_REMARKS_DATA_ENTRY;
```

```
ROUTINE EUPDATE
    [This routine is performed only when there are no errors committed on screen, it sorts screen
    log entries based on primarily OTS and then RTS times]

END EUPDATE;
```

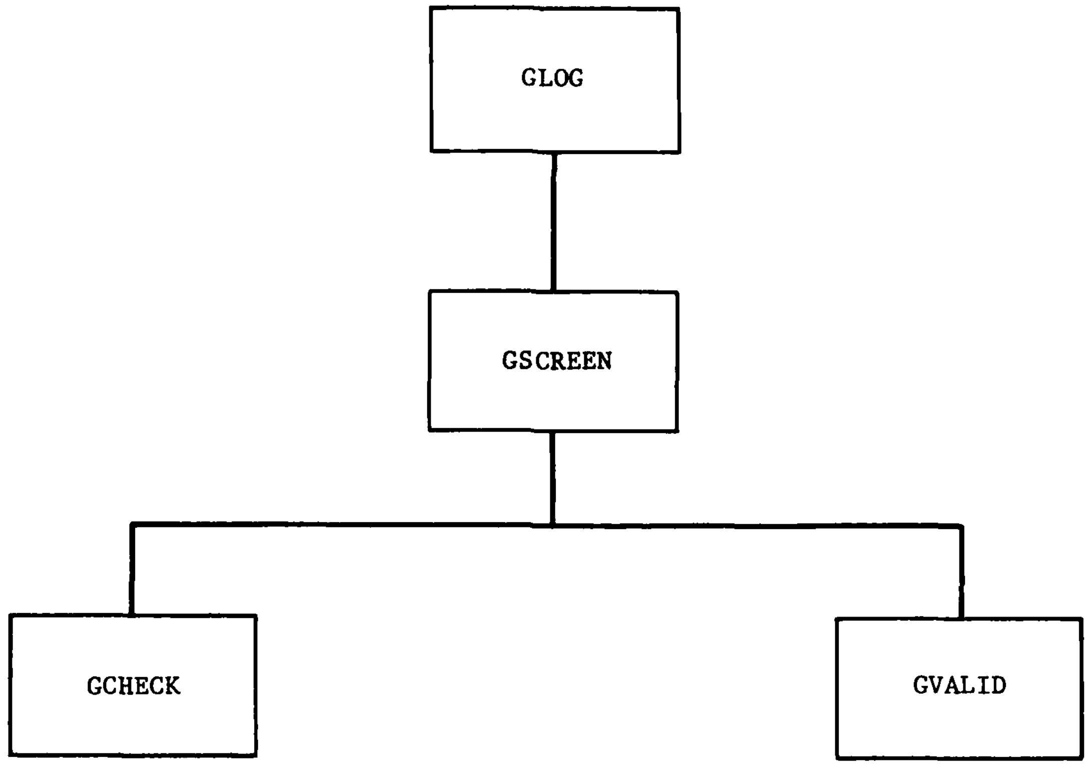


FIGURE A-7
SCHEMATIC DIAGRAM OF
DEMAND PLANNING LOG SCREEN ROUTINES

```
ROUTINE  GLOG
    [This routine invokes demand planning log screen]

    REPEAT UNTIL  (current program status is not equal to PFR);

        Save a copy of OAGLOG and CNVTOAG structures for 'restore previous screen' function;

        CALL  GSCREEN;
            [This routine controls demand planning log screen]

    ENDREPEAT;

    IF  screen message is equal to 'DATA ENTERED'

        THEN  update OAGLOG and CNVTOAG structures with new screen values;

END  GLOG;
```

```
ROUTINE  GSCREEN
    [This routine controls demand planning log screen]

    Set data masks to normal intensity;
    Set message data mask to high intensity;
    Set cursor to position 2; [2nd data field used by user]
    Set up pointers for initial, scroll, and screen message data fields;
         [to be used by DMS in loading and unloading data onto and from screen]

    Convert current time from character to numeric form;

    REPEAT UNTIL   (current program status is not equal to ENTER);

         Compute current hour;

         Set up screen pointers for four hours starting with current hour;

         PERFORM   DISPLAY_PANEL;

         IF   current program status is equal to PA1

              THEN  stop;

         IF   current program status is equal to ENTER

              THEN

                   Set data masks to normal intensity;

                   Set message data masks high intensity;

                   CALL  GCHECK;
                        [This routine checks for errors occured on screen as a result of an
                        erroneous entry and returns value for cursor pointing to first data field
                        where an error has occured; and an appropriate screen message is issued
                        advising user with corrections]

                   IF   screen message is not equal to 'DATA ENTERED'

                        THEN  highlight erroneous entry;
```

```
        ELSE

            CALL  GVALID;
                  [This routine performs data validation checks on screen entries
                  and returns value for cursor pointing to first invalide data
                  field.  Also, an appropriate screen message is issued advising
                  user with corrections]

        IF  screen message is not equal to 'DATA ENTERED'

            THEN  highlight invalid entry;

            ELSE  add current time to screen message;

    ENDREPEAT;

END  GSCREEN;
```

```
ROUTINE   GCHECK
    [This routine checks for errors occured on screen as a result of an erroneous entry and
    returns value for cursor pointing to first data field where an error has occured; and an
    appropriate screen message is issued advising user with corrections]

    IF   a character data is detected in any of numeric data fields

         THEN  issue message 'NUMERIC INPUT REQUIRED';

    IF   initial data field contains entry other than 'X ' or blank

         THEN  retrieve demand data from OAG data file;

    PERFORM   SCROOL_DATA_FIELD_ERROR_CHECK;

    EXITIF    error is detected

    REPEAT WHILE   (screen message is not equal to 'DATA ENTERED');
         [for each line of messages on screen]

         PERFORM   TTLARR_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   TTLDEP_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   KUBBS_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected

         Increment cursor;

         PERFORM   CGT_DATA_FIELD_ERROR_CHECK;

         EXITIF    error is detected
```

```
        Increment cursor;
        PERFORM   FARMM_DATA_FIELD_ERROR_CHECK;
        EXITIF    error is detected
        Increment cursor;
        PERFORM   NORTH_DATA_FIELD_ERROR_CHECK;
        EXITIF    error is detected
        Increment cursor;
        PERFORM   EAST_DATA_FIELD_ERROR_CHECK;
        EXITIF    error is detected
        Increment cursor;
        PERFORM   SOUTH_DATA_FIELD_ERROR_CHECK;
        EXITIF    error is detected
        Increment cursor;
        PERFORM   WEST_DATA_FIELD_ERROR_CHECK;
        EXITIF    error is detected
    ENDREPEAT;
END GCHECK;
```

```
PROCESS  SCROLL_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on scroll data field; if an error has occured, an appropriate
    message is issued]

END  SCROLL_DATA_FIELD_ERROR_CHECK;


PROCESS  TTLARR_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on total arrival demand data field; if an error has occured,
    an appropriate message is used]

END  TTLARR_DATA_FIELD_ERROR_CHECK;


PROCESS  TTLDEP_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on total departure demand data field; if an error has occured,
    an appropriate message is issued]

END  TTLDEP_DATA_FIELD_ERROR_CHECK;


PROCESS  KUBBS_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on KUBBS arrival demand data field; if an error has occured,
    an appropriate message is issued]

END  KUBBS_DATA_FIELD_ERROR_CHECK;


PROCESS  CGT_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on CGT arrival demand data field; if an error has occured, an
    appropriate message is issued]

END  CGT_DATA_FIELD_ERROR_CHECK;
```

```
PROCESS  VAINS_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on VAINS arrival demand data field; if an error has occured,
    an appropriate message is issued]

END  VAINS_DATA_FIELD_ERROR_CHECK;


PROCESS  FARMM_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on FARMM arrival demand data field; if an error has occured,
    an appropriate message is issued]

END  FARMM_DATA_FIELD_ERROR_CHECK;


PROCESS  NORTH_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on north departure demand data field; if an error has
    occurred, an appropriate message is issued]

END  NORTH_DATA_FIELD_ERROR_CHECK;


PROCESS  EAST_DATA_FIELD_ERROR_CHECK
    [This process errors on east departure demand data field; if an error has occured, an
    appropriate message is issued]

END  EAST_DATA_FIELD_ERROR_CHECK;


PROCESS  SOUTH_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on south departure demand data field; if an error has occured,
    an appropriate message is issued]

END  SOUTH_DATA_FIELD_ERROR_CHECK;
```

```
PROCESS  WEST_DATA_FIELD_ERROR_CHECK;
    [This process checks for errors on west departure demand data field; if an error has occured,
    an appropriate message is issued]

END  WEST_DATA_FIELD_ERROR_CHECK;
```

```
ROUTINE  GVALID
    [This routine performs data validation checks on screen entries and returns value to cursor
    pointing to first invalid data field.  Also, an appropriate screen message is issued advising
    user with corrections]

    REPEAT WHILE   (screen message is not equal to 'DATA ENTERED');
         [for each line of demand values on screen]

         IF   any of demand values as greater than 99

              THEN  issue message 'NUMBER OF AIRCRAFT MUST NOT EXCEED 99';

         IF   total arrival or departure demand entries are different than sum of individual
              arrival or departure demand

              THEN  issue message 'TOTAL DOES NOT EQUAL SUM OF INDIVIDUAL ENTRIES';

    ENDREPEAT;

END  GVALID;
```

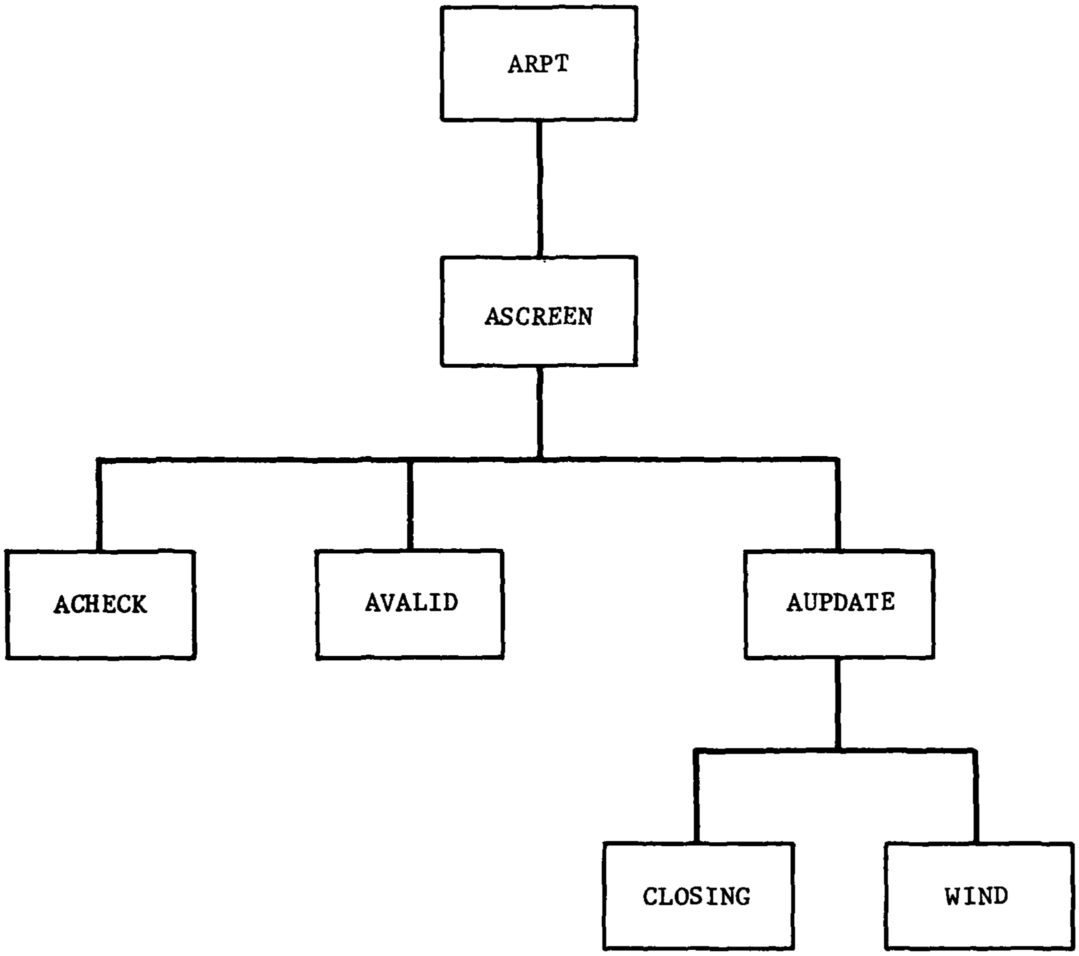


FIGURE A-8
SCHEMATIC DIAGRAM OF
AIRPORT STATUS SCREEN ROUTINES

```
ROUTINE  ARPT
    [This routine invokes airport status screen for both current and forecast environments]

    REPEAT UNTIL  (current program status is not equal to PF12);

        Save a copy of APTSTAT and CNVTAPT structures, and MIDFLAG variable for 'restore previous screen'
        function;

        REPEAT UNTIL  (current program status is not equal to PF3);

            Switch screens;  [current to forecast and vise versa]

            CALL  ASCREEN;
                [This routine controls airport status screen]

        ENDREPEAT;

    ENDREPEAT;

    IF  screen messages for both environments are equal to 'DATA ENTERED'

        THEN  update APTSTAT, MIDFLAG, and CNVTAPT with new screen entries;

END  ARPT;
```

```
ROUTINE  ASCREEN
    [This routine controls airport status screen]

    Set data masks to normal intensity;

    Set environment and message data masks to high intensity;

    Set cursor to position 2; [1st data field used by user]

    PERFORM  SET_UP_SCREEN_POINTERS_(ARPT);

    REPEAT UNTIL  (current program status is not equal to ENTER);

        PERFORM  DISPLAY_PANEL;

        IF  current program status is equal to PA1

            THEN  stop;

        IF  current program status is equal to ENTER

            THEN
                Set data masks to normal intensity;

                CALL  ACHECK;
                     [This routine checks for errors occurred on screen as a result of an erroneous entry
                     and returns value for cursor pointing to first data field where an error has
                     occurred; and an appropriate screen message is issued advising user with corrections]

                IF  screen message is not equal to 'DATA ENTERED'

                    THEN  highlight erroneous entry;

                    ELSE

                        CALL  AVALID;
                             [This routine performs data validation checks on screen entries and
                             returns value for cursor pointing to first invalid data field.  Also, an
                             appropriate screen message is issued advising user with corrections]
```

```
IF   Screen message is not equal to 'DATA ENTERED'
     THEN  highlight invalid entry;
     ELSE
          CALL  AUPDATE;
               [This routine performs local updates on screen]
          Add current time to screen message;
     ENDREPEAT;
END  ASCREEN;
```

```
PROCESS  SET_UP_SCREEN_POINTERS_(ARPT);
    [This process sets up screen pointers for DMS use]

    Structure AIRPORT_LOADLIST is set up with address of location of ARPT_DATA variables; [DMS uses
    AIRPORT_LOADLIST to load and unload data onto and from screen]

END  SET_UP_SCREEN_POINTERS_(ARPT);
```

```
ROUTINE  ACHECK
    [This routine checks for errors occurred on screen as a result of an erroneous entry and returns value
    for cursor pointing to first data field where an error has occurred; and an appropriate screen message
    is issued advising user with corrections]

    IF  a character data is detected in any of numeric data fields

         THEN  issue message 'NUMERIC INPUT REQUIRED';

    PERFORM  CEIL_DATA_FIELD_ERROR_CHECK;

    EXITIF  error is detected

    PERFORM  VIS_DATA_FIELD_ERROR_CHECK;

    EXITIF  error is detected

    PERFORM  WIND_DIR_DATA_FIELD_ERROR_CHECK;

    EXITIF  error is detected

    PERFORM  WIND_VEL_DATA_FIELD_ERROR_CHECK;

    EXITIF  error is detected

    IF  midflag data field contains entry other than 'X ' or blanks

         THEN  issue message 'INPUT MUST BE X OR BLANK';

         ELSE

              REPEAT WHILE  (screen message is equal to 'DATA ENTERED');  [for each runway]
                   Increment cursor;

                   IF   any of following data fields: tower imposed arrival and departure closure, runway
                        surface conditions and runway braking conditions entries contain any other entry
                        than 'X ' or blanks

                        THEN  issue message 'INPUT MUST BE X OR BLANK';

              ENDREPEAT;

END  ACHECK;
```

```
PROCESS  CEIL_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on ceiling data field; if an error has occurred, an appropriate message
    is issued;

END  CEIL_DATA_FIELD_ERROR_CHECK;


PROCESS  VIS_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on visibility data field; if an error has occurred, an appropriate
    message is issued;

END  VIS_DATA_FIELD_ERROR_CHECK;


PROCESS  WIND_DIR_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on wind direction data field; if an error has occurred, an appropriate
    message is issued;

END  WIND_DIR_DATA_FIELD_ERROR_CHECK;


PROCESS  WIND_VEL_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on wind velocity data field; if an error has occurred, an appropriate
    message is issued;

END  WIND_VEL_DATA_FIELD_ERROR_CHECK;
```

```
ROUTINE  AVALID
    [This routine performs data validation checks on screen entries and returns value for cursor pointing to
    first invalid data field.  Also, an appropriate screen message is issued advising user with corrections]

    IF  airport visibility is greater than 100

        THEN  issue message 'VISIBILITY MUST BE LESS THAN 100 MILES';

    IF  wind direction is greater than 360

        THEN  issue message 'WIND DIRECTION MUST BE LESS THAN 360 DEGREES';

    REPEAT WHILE  (screen message is equal to 'DATA ENTERED');  [for each runway]

        IF  surface condition is dry AND braking condition is poor

        THEN  issue message 'SURFACE AND BRAKING CONDITIONS ARE NOT CONSISTENT';

    ENDREPEAT;

END  AVALID;


ROUTINE  AUPDATE
    [This routine performs local updates on screen]

    CALL  WIND;
         [This routine computes crosswind and tailwind components of prevailing wind and sets up
         corresponding screen data fields]

    CALL  CLOSING;
         [This routine closes runways based on wind direction and weather minima]

END  AUPDATE;
```

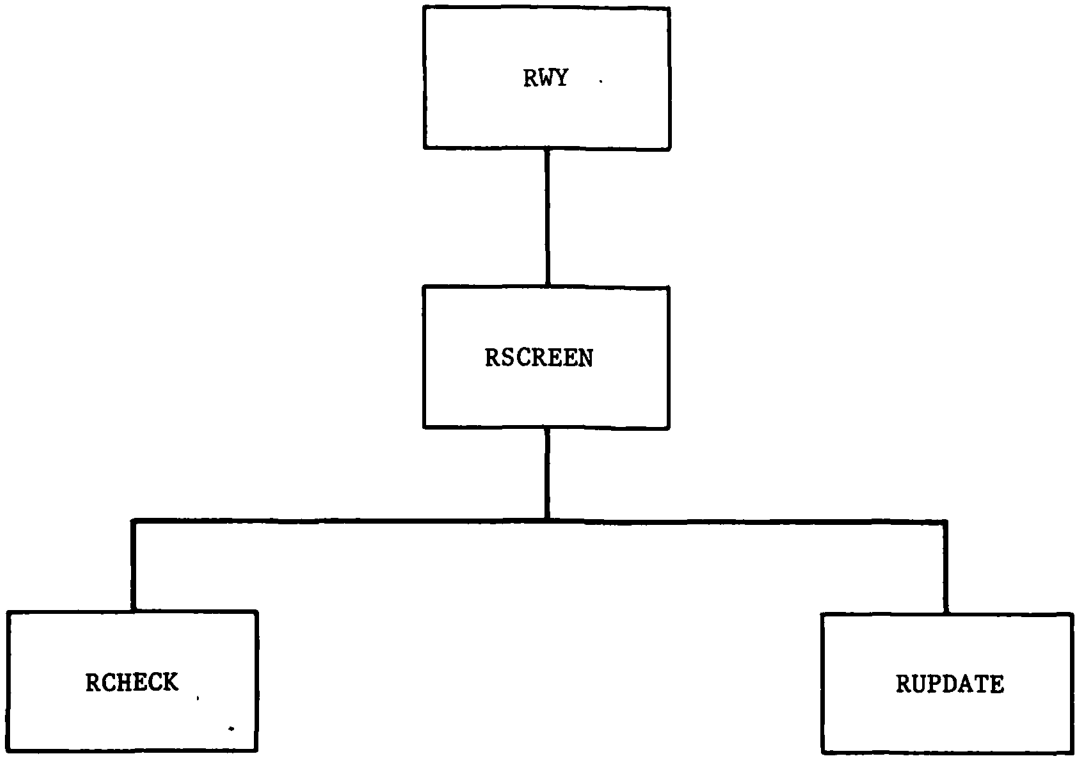


FIGURE A-9
SCHEMATIC DIAGRAM OF
EQUIPMENT STATUS SCREEN ROUTINES

```
ROUTINE  RWY
    [This routine invokes runway equipment status screen for both current and forecast environment]

    REPEAT UNTIL   (current program status is not equal to PF12);

         Save a copy of RWYEQP structure for 'restore original screen' function;

         REPEAT UNTIL   (current program status is not equal to PF4);

              Switch screens;  [current to forecast or vise versa]

              CALL  RSCREEN;
                   [This routine controls runway equipment status screen]

         ENDREPEAT;

    ENDREPEAT;

    IF   screen messages for both environments are equal to 'DATA ENTERED'

         THEN  update RWYEQP with new screen entries;

END  RWY;
```

```
ROUTINE  RSCREEN
    [This routine controls runway equipment status screen]

    Set data masks to normal intensity;

    Set environment and message data masks to high intensity;

    Set cursor to position 3;  [1st data field used by user]

    PERFORM   SET_UP_SCREEN_POINTERS_(RWY);

    REPEAT UNTIL   (current program status is not equal to ENTER);

        PERFORM   DISPLAY_PANEL;

        IF   current program status is equal to PA1

             THEN  stop;

        IF  current program status is equal to ENTER

             THEN
                  Set data masks to normal intensity;

                  CALL  RCHECK;
                       [This routine checks for errors occured on screen as a result of erroneous
                       entry and returns value for cursor pointing to first data field where an
                       error has occured; and an appropriate screen message is issued advising
                       user with corrections]

                  IF   screen message is not equal to 'DATA ENTERED'

                       THEN  highlight erroneous entry;

                       ELSE

                           CALL  RUPDATE;
                                [This routine performs local updates on screen]

                           Add current time to screen message;

    ENDREPEAT;

END  RSCREEN;
```

<u>PROCESS</u> SET_UP_SCREEN_POINTERS_(RWY)
[This process sets up screen pointers for DMS use]

Structure RWY_LOADLIST is set up with address of location of RWY_DATA variable; [DMS uses RWY LOADLIST to load and unload data onto and from screen]

<u>END</u> SET_UP_SCREEN_POINTERS_(RWY`;

```
ROUTINE  RCHECK
    [This routine checks for errors occured on screen as a result of an erroneous entry and
    returns value for cursor pointing to first data field where an error has occured, and an
    appropriate screen message is issued advising user with corrections]

    REPEAT WHILE   (screen message is equal to 'DATA ENTERED');
                            [for each runway]

        IF   entry for each equipment type available for a runway is not equal to blank or 'x '

             THEN      issue message 'INPUT MUST BE X OR BLANK';

    ENDREPEAT;

END  RCHECK;


ROUTINE  RUPDATE
    [This routine performs local updates on screen]

    REPEAT; (for runways 14R and 14L)

        IF   any of following equipment:  localizer, glide slope, outer marker, inner marker,
             middle marker, RVR, ALS, HIRL, CL, or TDZ is inoperable

             THEN  Set CATII inoperable;

    ENDREPEAT;

END  RUPDATE;
```

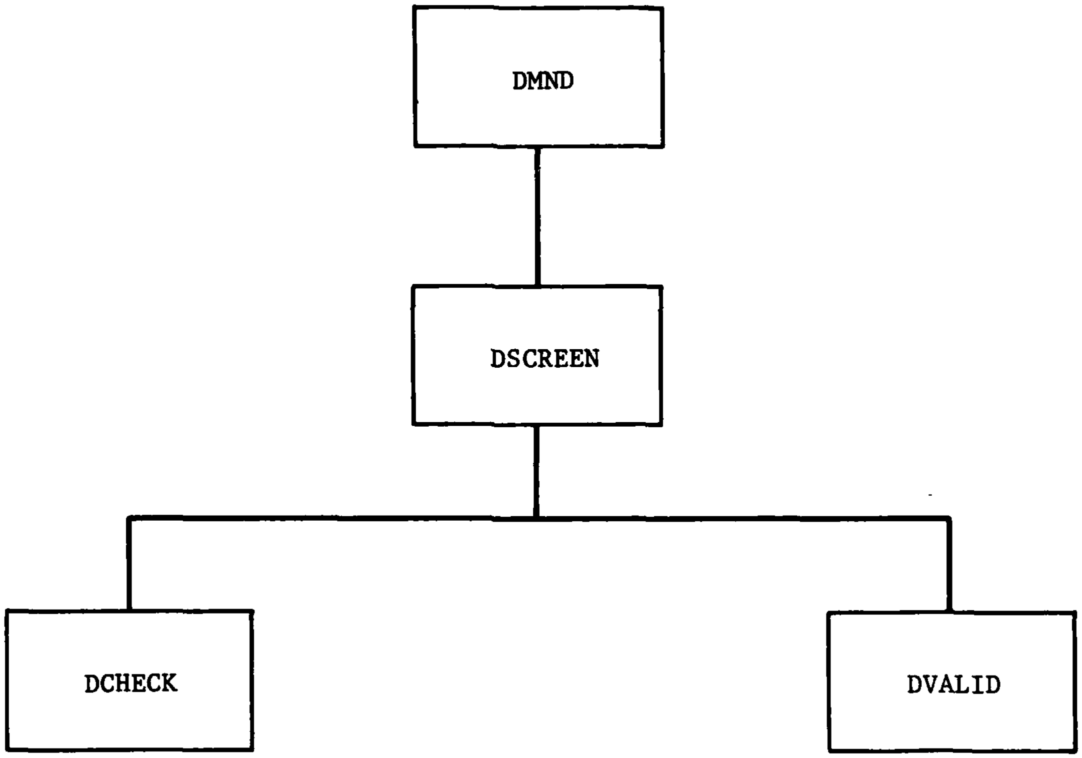


FIGURE A-10
SCHEMATIC DIAGRAM OF
DEMAND PROFILE SCREEN ROUTINES

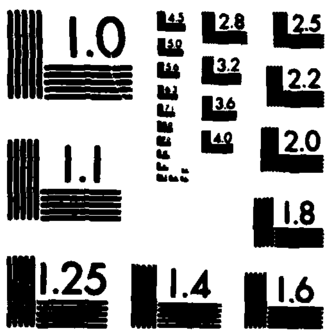
```
ROUTINE  DMND
    [This routine invokes demand profile screen for both current and forecast environments]

    REPEAT UNTIL   (current program status is not equal to PF12);

         Save a copy of DEMAND and CNVTDEM structures for 'restore previous screen' function;

         REPEAT UNTIL   (current program status is not equal to PF5);

              Switch screens; [current to forecast and vise versa]

              CALL  DSCREEN;

                   [This routine controls demand profile screen]

         ENDREPEAT;

    ENDREPEAT;

    IF   screen messages for both environments are equal to 'DATA ENTERED'

         THEN update DEMAND and CNVTDEM with new screen entries;

END  DMND;
```

```
ROUTINE  DSCREEN
    [This routine controls demand profile screen]

    Set data masks to normal intensity;
    Set environment and message data masks to high intensity;
    Set cursor to position 4;
    Determine current time;

    PERFORM   SET_UP_SCREEN_POINTERS_(DMND);

    REPEAT UNTIL   (current program status is not equal to ENTER);

         PERFORM   DISPLAY_PANEL;

         IF   current program status is equal to PA1

              THEN stop;

         IF   current program status is equal to ENTER

              THEN
                   Set data masks to normal intensity;

                   CALL  DCHECK;
                        [This routine checks for errors occurred on screen as a result of an
                        erroneous entry and returns value for cursor pointing to first data field
                        where an error has occurred; and an appropriate screen message is issued
                        advising user with corrections]

                   IF   screen message is not equal to 'DATA ENTERED'

                        THEN  highlight erroneous entry;

                        ELSE

                             CALL  DVALID;
                                  [This routine performs data validation checks on screen entries
                                  and returns value for cursor pointing to first invalid data
                                  field.  Also, an appropriate screen message is issued advising
                                  user with corrections]
```

```
            IF   screen message is not equal to 'DATA ENTERED'
                 THEN  highlight invalid entry;
                 ELSI
                      Add current time to screen message;
    ENDREPEAT;
END  DSCREEN;
```

AD-A127 398 SOFTWARE DESCRIPTION FOR THE O'HARE RUNWAY CONFIGURATION MANAGEMENT SYSTE..(U) MITRE CORP MCLEAN VA METREK DIV S KAVOUSSI OCT 82 MTR-82W125-VOL-1 3/3

UNCLASSIFIED FAA-EM-82-28-VOL-1 DTFA01-81-C-10003 F/G 1/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
PROCESS  SET_UP_SCREEN_POINTERS_(DMND)
    [This process sets up screen pointers for DMS use]

    Structure DMND_LOADLIST is set up with address of location of DMND_DATA variables; [DMS uses
    DMND_LOADLIST to load and unload data onto and from screen]

END SET_UP_SCREEN_POINTERS_(DMND);
```

ROUTINE DCHECK

[This routine checks for errors occurred on screen as a result of an erroneous entry and returns value for cursor pointing to first data field where an error has occurred; and an appropriate screen message is issued advising user with corrections]

IF a character data is detected in any of numeric data fields

THEN issue message 'NUMERIC INPUT REQUIRED';

IF RETRIEVE entry is not equal to blank or 'X '

THEN issue message 'INPUT MUST BE X OR BLANK;

Increment cursor;

PERFORM ARR_TOTAL_DATA_FIELD_ERROR_CHECK;

EXITIF error is detected

Increment cursor;

PERFORM ARR_KUBBS_DATA_FIELD_ERROR_CHECK;

EXITIF error is detected

Increment cursor;

PERFORM ARR_CGT_DATA_FIELD_ERROR_CHECK;

EXITIF error is detected

Increment cursor;

PERFORM ARR_VAINS_DATA_FIELD_CHECK;

EXITIF error is detected

Increment cursor;

PERFORM ARR_FARMM_DATA_FIELD_ERROR_CHECK;

```
    EXITIF    error is detected
    Increment cursor;
    PERFORM   DEP_TOTAL_DATA_FIELD_ERROR_CHECK;
    EXITIF    error is detected
    Increment cursor;
    PERFORM   DEP_NORTH_DATA_FIELD_ERROR_CHECK;
    EXITIF    error is detected
    Increment cursor;
    PERFORM   DEP_EAST_DATA_FIELD_ERROR_CHECK;
    EXITIF    error is detected
    Increment cursor;
    PERFORM   DEP_SOUTH_DATA_FIELD_ERROR_CHECK;
    EXITIF    error is detected
    Increment cursor;
    PERFORM   DEP_WEST_DATA_FIELD_ERROR_CHECK;
END DCHECK;
```

```
PROCESS  ARR_TOTAL_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on total arrival demand data field; if an error is detected,
    an appropriate message is issued]

END  ARR_TOTAL_DATA_FIELD_ERROR_CHECK;


PROCESS  ARR_KUBBS_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on fix KUBBS arrival demand data field; if an error is
    detected, an appropriate message is issued]

END  ARR_KUBBS_DATA_FIELD_ERROR_CHECK;


PROCESS  ARR_CGT_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on fix CGT arrival demand data field; if an error is detected,
    an appropriate message is issued]

END  ARR_CGT_DATA_FIELD_ERROR_CHECK;


PROCESS  ARR_VAINS_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on fix VAINS arrival demand data field; if an error is
    detected, an appropriate message is issued]

END  ARR_VAINS_DATA_FIELD_ERROR_CHECK;


PROCESS  ARR_FARMM_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on fix FARMM arrival demand data field; if an error is
    detected, an appropriate message is issued]

END  ARR_FARMM_DATA_FIELD_ERROR_CHECK;
```

```
PROCESS  ARR_DEP_TOTAL_FIELD_ERROR_CHECK
    [This process checks for errors on total departure demand data field; if an error is detected,
    an appropriate message is issued]

END  ARR_FARMM_DATA_FIELD_ERROR_CHECK;


PROCESS  DEP_NORTH_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on NORTH fix departure demand data field; if an error is
    detected, an appropriate message is issued]

END  DEP_NORTH_DATA_FIELD_ERROR_CHECK;


PROCESS  DEP_EAST_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on EAST fix departure demand data field; if an error is
    detected, an appropriate message is issued]

END  DEP_EAST_DATA_FIELD_ERROR_CHECK;


PROCESS  DEP_SOUTH_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on SOUTH fix departure demand data field; if an error is
    detected, an appropriate message is issued]

END  DEP_SOUTH_DATA_FIELD_ERROR_CHECK;


PROCESS  DEP_WEST_DATA_FIELD_ERROR_CHECK
    [This process checks for errors on WEST fix departure demand data field; if an error is
    detected, an appropriate message is issued]

END  DEP_WEST_DATA_FIELD_ERROR_CHECK;
```

```
ROUTINE  DVALID
    [This routine performs data validation checks on screen entries and returns value for cursor
    pointing to first invalid data field.  Also, an appropriate screen message is issued advising
    user with corrections]

    IF   RETRIEVE entry is not equal to blank

         THEN PERFORM   RETRIEVE_DEMAND_DATA_FROM_DEMAND_LOG;

    REPEAT; (for each demand entry on screen)

         IF   demand entry other than total entries is greater than 99

              THEN  issue message 'NUMBER OF AIRCRAFT MUST NOT EXCEED 99';

         IF   total arrival or departure demand entries is different than sum of individual
              arrival or departure demands

              THEN  issue message 'TOTAL DOES NOT EQUAL SUM OF INDIVIDUAL ENTRIES';

    ENDREPEAT;

END  DVALID;
```

__PROCESS__ RETRIEVE_DEMAND_DATA_FROM_DEMAND_LOG

[This process extracts demand data from demand planning log]

__END__ RETRIEVE_DEMAND_DATA_FROM_DEMAND_LOG;

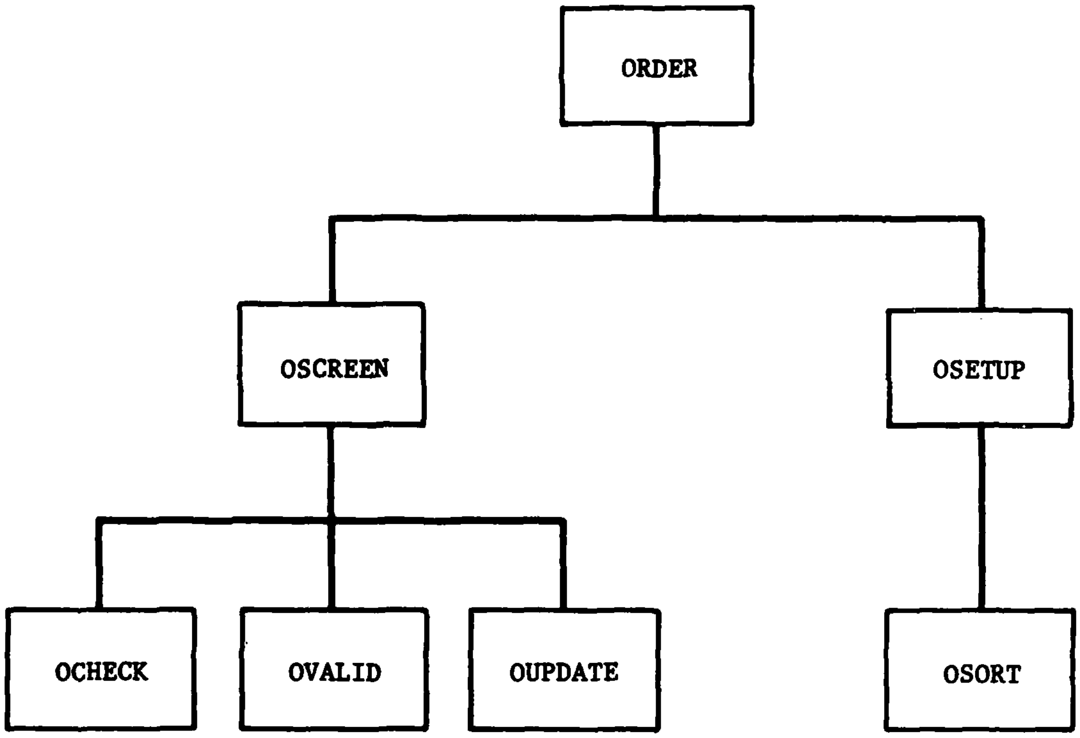


FIGURE A-11
SCHEMATIC DIAGRAM OF
ORDERED LIST OF CONFIGURATIONS SCREEN ROUTINES

```
ROUTINE  ORDER
    [This routine invokes ordered list of configurations screen for both current and forecast environments]

    CALL  OSETUP;
         [This routine sets up information on ordered list of configurations screen]

    REPEAT UNTIL  (current program status is not equal to PF12);
         Save a copy of screen message and current operating configuraton index for 'restore previous
         screen' function;

         REPEAT UNTIL  (current program status is not equal to PF6);
              Switch screens;  [current to forecast or vise versa]

              CALL  OSCREEN;
                   [This routine controls ordered list of configurations screen]

         ENDREPEAT;

    ENDREPEAT;

    IF  screen messages for both environments are equal to 'DATA ENTERED'

         THEN  update current operating configuration's indices with new screen entries;

END  ORDER;
```

```
ROUTINE  OSETUP
    [This routine sets up information on ordered list of configurations screen]

    REPEAT; (for each environment)

        Set capacity numbers and associated configuration indices from INFORM structure;
        Compute percentage of arrivals;
        Set percentage of arrivals;

        CALL  OSORT;
            [This routine sorts list of configurations based on capacity]

        REPEAT; (for each configuration)

            IF  configuration is eligible

                THEN

                    PERFORM  SCREEN_PARAMETERS_SET_UP;

                    PERFORM  FLAG_SETTING;

        ENDREPEAT;

        Set number of eligible configurations;

    ENDREPEAT;

END  OSETUP;
```

```
PROCESS  SCREEN_PARAMETERS_SET_UP
    [This process sets up parameters on ordered list of configurations screen]

    Set following parameters:  arrival runways ID, departure runways ID, and configuration capacity;

END  SCREEN_PARAMETERS_SET_UP;


PROCESS  FLAG_SETTING
    [This process determines warning flags for ordered list of configurations screen]

    IF a configuration contains runways with HIRL inoperable

        THEN  set flag 'DAY ONLY';

    IF  demand exceeds capacity of configuration

        THEN  set flag 'SATURATED';

    IF  Midway indicator is not blank

        THEN  set flag 'MIDWAY';

END  FLAG_SETTING;
```

```
ROUTINE OSORT
    [This routine sorts configurations list based on capacity; Shell's method is used]

END OSORT;
```

```
ROUTINE  OSCREEN
    [This routine controls ordered list of configurations screen]

    PERFORM  SET_UP_SCREEN_PERMANENT_POINTERS_(ORDER);

    PERFORM  SCREEN_PROGRAM_INITIALIZATION;

    REPEAT UNTIL  (current program status is not equal to ENTER);

        PERFORM  SCREEN_SCROLL;

        PERFORM  DISPLAY_PANEL;

        IF  current program status is equal to PA1

            THEN  stop;

        IF  current program status is equal to ENTER

            THEN
              Set data masks to normal intensity;

              CALL  OCHECK;
                  [This routine checks for errors occurred on screen as a result of an erroneous entry
                  and returns value for cursor pointing to first data field where an error has
                  occurred; and an appropriate message is issued advising user with corrections]

              IF  screen message is not equal to 'DATA ENTERED'

                  THEN  highlight erroneous entry;

                  ELSE

                      CALL  OVALID;
                          [This routine performs data validation checks on screen entries and
                          returns value for cursor pointing to first invalid data field.  Also, an
                          appropriate screen message is issued advising user with corrections]
```

```
            IF  screen message is not equal to 'DATA ENTERED'

                THEN  highlight invalid entry;

                ELSE

                    CALL  OUPDATE;
                        [This routine updates configuration index parameter locally]

                    Add current time to screen message;

    ENDREPEAT;

END  OSCREEN;
```

```
PROCESS  SET_UP_SCREEN_PERMANENT_POINTERS_(ORDER)
    [This process sets up screen pointers for permanent screen variables for DMS use]

    Structure ORDER_LOADLIST is set up with address of location of CONFLST variables;
    [DMS uses ORDER_LOADLIST to load and unload data onto and from screen]

END  SET_UP_SCREEN_PERMANENT_POINTERS_(ORDER);


PROCESS  SCREEN_PROGRAM_INITIALIZATION
    [This process performs a number of variable initializations for screen routine]

    IF  all configurations are ineligible

        THEN  issue message 'NO ELIGIBLE CONFIGURATION';

    Set scrolling function parameters;

    Set select entry to blank;

END  SCREEN_PROGRAM_INITIALIZATION;


PROCESS  SCREEN_SCROLL
    [This process performs scrolling function for ordered list of configurations screen by setting pointers
    to first and last lines of data to appear on screen at one time]

    Add value of scroll data field to pointers signifying first and last line of data to appear on screen;
    (negative scroll numbers are allowed)

    Perform bookkeeping to determine how many lines of data are to be displayed and their locations on list
    of configurations;

    Set up screen pointers for configurations to be displayed

    Highlight current operating configuration if it appears on screen;

    Darken unused portion of screen;

END  SCREEN_SCROLL;
```

```
ROUTINE  OCHECK
   [This routine checks for errors occurred on screen as a result of an erroneous entry and returns value
   for cursor pointing to first data field where an error has occurred; and an appropriate screen message
   is issued advising user with corrections]

   IF  a character data is detected in numeric data field (scroll data field)

        THEN  issue message 'NUMERIC INPUT REQUIRED'

   IF  a decimal point is detected in scroll data field

        THEN  issue message 'NO DECIMAL POINTS ALLOWED;

   REPEAT; (for every line of data on screen)

        IF  select data field contains an entry other than 'X ' or blank

             THEN  issue message 'INPUT MUST BE X OR BLANK';

   ENDREPEAT;

END  OCHECK;


ROUTINE  OVALID
   [This routine performs data validation checks on screen entries and returns value for cursor pointing to
   first invalid data field.  Also, an appropriate screen message is issued advising user with corrections]

   IF  more than one select data field is not equal to blank

        THEN  issue message 'SELECT ONLY ONE CONFIGURATION';

END  OVALID;


ROUTINE  OUPDATE
   [This routine updates configuration index parameter locally]

END  OUPDATE;
```

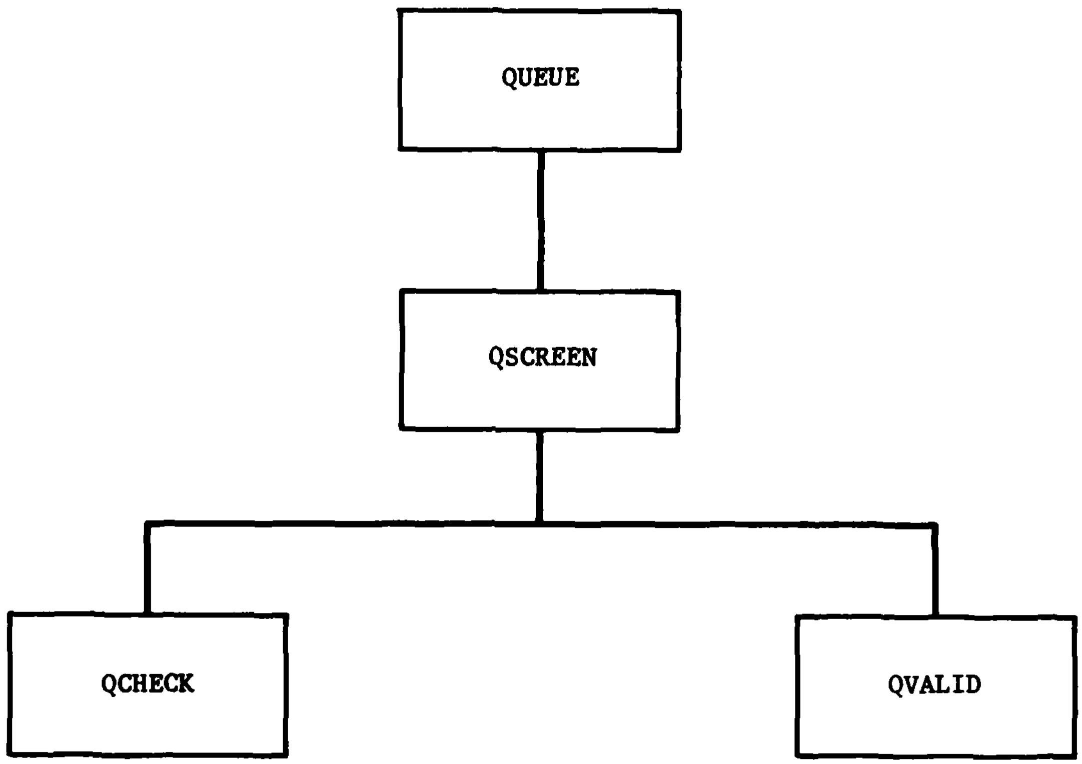


FIGURE A-12
SCHEMATIC DIAGRAM OF
DEPARTURE QUEUE SCREEN ROUTINES

```
ROUTINE QUEUE
    [This routine invokes current departure queue screen]

    REPEAT UNTIL (current program status is not equal to PF12);

        Save a copy of QUELEN and CNVTQLN structures for 'restore previous screen' function;

        REPEAT UNTIL (current program status is not equal to PF7);

            CALL QSCREEN;
                [This routine controls current departure queue screen]

        ENDREPEAT;

    ENDREPEAT;

    IF screen message is equal to 'DATA ENTERED'

        THEN update QUELEN and CNVTQLN with new screen entries;

END QUEUE;
```

```
ROUTINE  QSCREEN
    [This routine controls current departure queue screen]

    Set data masks to normal intensity;

    Set environment and message data masks to high intensity;

    Set cursor to position 2;

    PERFORM  SET_UP_SCREEN_POINTERS_(QUEUE);

    Determine number of departure runways in current configuration;

    IF  number of departure runways are less than 4

        THEN  darken part of screen for excess departure runways;

    REPEAT UNTIL  (current program status is not equal to ENTER);

        PERFORM  DISPLAY_PANEL;

        IF  current program status is equal to PA1

            THEN  stop;

        IF  current program status is equal to ENTER

            THEN
                Set data masks to normal intensity;

                CALL  QCHECK;
                    [This routine checks for errors occurred on screen as a result of an erroneous entry
                     and returns value for cursor pointing to first data field where an error has
                     occurred; and an appropriate screen message is issued advising user with corrections]

                IF  screen message is not equal to 'DATA ENTERED'

                    THEN  highlight erroneous entry;
```

```
            ELSE

                CALL  QVALID;
                      [This routine right-justifies data on screen]

                Add current time to screen message;

      ENDREPEAT;

  END  QSCREEN;
```

```
PROCESS  SET_UP_SCREEN_POINTERS_(QUEUE);
    [This process sets up screen pointers for DMS use].

    Structure QUE_LOADLIST is set up with address of location of QUELEN_DATA variables;
    [DMS uses QUE_LOADLIST to load and unload data onto and from screen]

END  SET_UP_SCREEN_POINTERS_(QUEUE);
```

```
ROUTINE  QCHECK
    [This routine checks for errors occurred on screen as a result of an erroneous entry and returns value
    for cursor pointing to first data field where an error has occurred; and an appropriate screen message
    is issued advising user with corrections]

    IF  a character data is detected in any of numeric data fields

        THEN  issue message 'NUMERIC INPUT REQUIRED';

    IF  a decimal point is detected in any of data fields

        THEN  issue message 'NO DECIMAL POINTS ALLOWED';

    IF  a negative sign is detected in any of data fields

        THEN  issue message 'NON-NEGATIVE INPUT REQUIRED';

END  QCHECK;


ROUTINE  QVALID
    [This routine right-justifies data on screen]

END  QVALID;
```

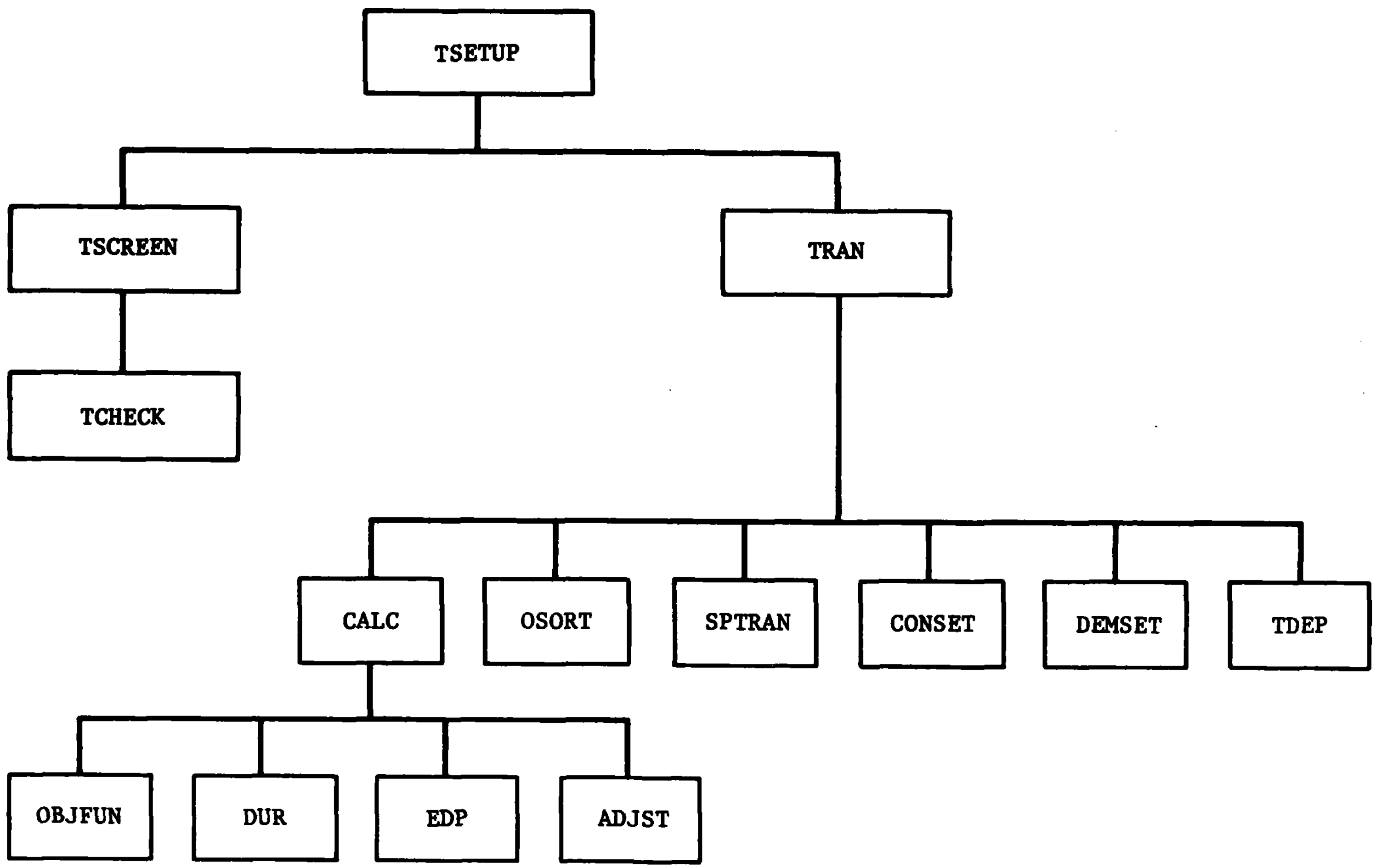


FIGURE A-13
SCHEMATIC DIAGRAM OF
ORDERED LIST OF TRANSITIONS SCREEN ROUTINES

```
ROUTINE  TSETUP
    [This routine invokes ordered list of transitions screen]

    CALL  TRAN;
         [This routine performs transition computation and transition screen parameter set up]

    REPEAT UNTIL  (current program status is not equal to PF12);

         Save a copy screen message and scroll data field value;

         REPEAT UNTIL  (current program status is not equal to PF8);

              CALL  TSCREEN;
                   [This routine controls ordered list of transitions screen]

         ENDREPEAT;

    ENDREPEAT;

END  TSETUP;
```

```
ROUTINE  TSCREEN
  [This routine controls ordered list of transitions screen]

  PERFORM  SET_UP_SCREEN_PERMANENT_POINTERS_(TSETUP);

  PERFORM  SCREEN_PROGRAM_INITIALIZATION;

  IF  all configurations in forecast environment are ineligible

      THEN  issue message 'NO ELIGIBLE CONFIGURATIONS';

  REPEAT UNTIL  (current program status is not equal to ENTER);

      PERFORM  SCREEN_SCROLL;

      PERFORM  DISPLAY_PANEL;

      IF  current program status is equal to PA1

          THEN  stop;

      IF  current program status is equal to ENTER

          THEN

              Set data masks to normal intensity;

              Set current operating configuration and screen message data masks to high intensity;

              CALL  TCHECK;
                  [This routine checks for errors occurred on screen as a result of an erroneous entry
                  and returns value for cursor pointing to first data field where an error has
                  occurred; and an appropriate screen message is issued advising user with corrections]

                  IF  screen message is not equal to 'DATA ENTERED'

                      THEN  highlight erroneous entry;

                      ELSE  add current time to screen message;

      ENDREPEAT;

END  TSCREEN;
```

PROCESS SET_UP_SCREEN_PERMANENT_POINTERS (TSETUP)
[This process sets up screen pointers of permanent variables for DMS use]

Structure TRANS_LOADLIST is set up with address of location of TRANLST variables; [DMS uses TRANS_LOADLIST to load and unload data onto and from screen]

END SET_UP_SCREEN_PERMANENT_POINTERS_(TSETUP);

PROCESS SCREEN_PROGRAM_INITIALIZATION
[This process performs a number of variable initializations for screen routine]

Set data masks to normal intensity;

Set current operating configuration and screen message data masks to high intensity;

Set scrolling function parameters;

END SCREEN_PROGRAM_INITIALIZATION;

PROCESS SCREEN_SCROLL
[This process performs scrolling function for ordered list of transitions screen by setting pointers to first and last lines of data to appear on screen at one time]

Add value of scroll data field to pointers signifying first and last line of data to appear on screen;
[negative scroll numbers are allowed]

Perform bookkeeping to determine how many line of data are to be displayed and their locations on list of transitions;

Set up screen pointers for configurations to be displayed on screen;

Darken unusued portion of screen;

END SCREEN_SCROLL;

```
ROUTINE  TCHECK
    [This routine checks for errors occurred on screen as a result of an erroneous entry and returns value
    for cursor pointing to first data field where an error has occurred; and an appropriate screen message
    is issued advising user with corrections]

    IF  a character data is detected in scroll data field

        THEN  issue message 'NUMERIC INPUT REQUIRED';

    IF  a decimal point is detected in scroll data field

        THEN  issue message 'NO DECIMAL POINTS ALLOWED';

END  TCHECK;
```

```
ROUTINE  TRAN
    [this routine performs transition computations and transition screen parameters setup]

    Compute percentage of arrivals for forecast environment;

    IF  current configuration is eligible

        THEN

            CALL  CONSET;  [uses current configuration]
                [This routine sets variable CONFIGDATA which signifies runways in a configuration]

            CALL  DEMSET;
                [This routine computes demand values for each fix pertaining to current configuration]

            Determine number of LP variables pertaining to current configuration;

            Set up variable INT with time required for queue flush out - for each departure runway based
            on departure queue length;

        REPEAT; (for each configuration)

            IF   a configuration is eligible in forecast environment and it is not the same as current
                 configuration

                 THEN

                     CALL  CONSET;  [uses forecast configuration]
                         [This routine sets variable CONFIGDATA which signifies runways in a
                         configuration]

                         Determine number of LP variables pertaining to forecast configuration;

                     IF   transition is from IFR to VFR and current configuration will be ineligible in
                          forecast conditions

                          THEN

                              CALL  DEMSET;
                                  [This routine computes demand values for each fix pertaining to final
                                  configuration]
```

```
    CALL  SPTRAN;
          [Special routine to compute transition capacity bypassing LP
          algorithm]

ELSEIF    transition is taking place in same environment, i.e., VFR to VFR or
          IFR to IFR and both configurations are eligible

    THEN

          CALL  DEMSET;
                [This routine computes demand values for each fix pertaining to
                final configuration]

          CALL  TDEP;
                [This routine prepares dependence matrix]

          CALL  CALC;
                [This routine performs LP algorithm and determines transition
                duration and capacity]

          ELSEIF  transition is from IFR to VFR

                THEN

                      CALL  DEMSET;
                            [This routine computes demand values for each fix
                            pertaining to final configuration]

                      CALL  SPTRAN;
                            [Special routine to compute transition capacity,
                            bypassing LP algorithm]

          ELSEIF    transition is from VFR to IFR and current configuration is
                    eligible in forecast environment

                THEN

                      CALL  DEMSET;
                            [This routine computes demand values for each fix
                            pertaining to current configuration]
```

```
                        CALL  TDEP;
                              [This routine prepares dependence matrix]

                        CALL  CALC;
                              [This routine performs LP algorithm and determines
                              transition duration and capacity]

                   ELSE  for all remaining cases

                        CALL  DEMSET;
                              [This routine computes demand values for each fix
                              pertaining to final configuration]

                        CALL  SPTRAN;
                              [Special routine to compute transition capacity
                              bypassing LP algorithm]

          Set transition duration;

          Set final configuration's capacity in forecast environment;

          Compute transition hour capacity by prorating final capacity for remaining portion of
          hour added to transition capacity;

          Set transition hour capacity;

      ELSE  (final configuration is ineligible in forecast environment);

    ENDREPEAT;

  CALL  OSORT;
    [This routine sorts list of transitions on transition hour capacity]

    REPEAT; (for each transition)

    Set screen parameter;

    ENDREPEAT;

END  TRAN;
```

```
ROUTINE CONSET
    [This routine determines variable CONFIGDATA which signifies runways in a configuration]

END CONSET;


ROUTINE DEMSET
    [This routine computes load at fixes for current and forecast configurations]

    Set load at fixes equal to arrival capacity multiplied by demand at fix divided by total arrival demand;

END DEMSET;


ROUTINE TDEP
    [This routine computes dependence matrix for two configurations involved in transition from dependence
    matrix data file]

END TDEP;
```

```
ROUTINE SPTRAN
    [This routine performs a special transition algorithm in case when two configurations are not mutually
    eligible]

    Compute contribution of current configuration to transition duration; (maximum of travel times)

    Include effect of departure queues if applicable;

    Compute contribution of final configuration to transition duration; (maximum of travel times)

    Compute transition duration by taking maximum of contributions of current and final configurations
    calculated above;

    IF transition duration is driven from current configuration

        THEN

            Compute capacity assuming all fixes stop feeding current configuration at same time; (capacity
            computed by multiplying fix loads by duration of time each fix is active)

        ELSEIF transition duration is driven from final configuration

            THEN
                Compute capacity assuming all fixes stop feeding current configuration at same time plus
                capacity computed during remainder of transition duration from final configuration
                operating;

END SPTRAN;
```

```
ROUTINE CALC
    [This routine computes LP solution using special algorithm, also transition duration is computed]

   IF arrival runways in both configurations in transition are identical

      THEN CALL ADJST;
         [If both configurations in transition have same arrival runways then routine ADJST uses a
         different algorithm to compute transition duration and capacity]

      ELSE

         CALL DUR;
            [This routine computes transition duration]

         Modify transition duration with information on current departure queues;

         CALL EDP;
            [This routine prepares expanded dependence matrix]

         Compute upperbound constraints for variables pertaining to final configuration;

         Compute upperbound constraints for variables pertaining to current configuration;

         Determine initial solution for LP variables;

         PERFORM LP_ALGORITHM_ITERATIONS;

      Compute transition capacity using LP solution and transition duration via OBJFUN function;

END CALC;
```

```
PROCESS  LP_ALGORITHM_ITERATION;
    [This process performs LP algorithm's iterations]

REPEAT UNTIL  (no improvement on objective function is possible)

    Construct a matrix of zeroes and ones (ALT);

        [1 - if an equality constraint between Jth variable of final configuration and ith variable of
        current configuration exist]

    Construct bit strings from columns (COL) and rows (ROW) of matrix ALT;

    Determine connected groupings of entries 'one' of matrix ALT;

    Compute sum of cost coefficients of variables pertaining to current configuration (along rows of matrix
    ALT) within each group;

    Compute sum of cost coefficient of variables pertaining to final configuration (along columns of matrix
    ALT) within each group;

    IF   sum of cost coefficients of variables in final configuration is less than sum of cost coefficients
         of variables in final configuration within each group

         THEN  [improvement in objective function is possible]

              Adjust LP variables within each group;

    ENDREPEAT;

END  LP_ALGORITHM_ITERATION;
```

ROUTINE DUR
[This routine computes transition duration]

Compute contribution of current configuration to transition duration; [maximum of travel times between fixes and runways]

Add contribution of current configuration to maximum delay extracted from exclusive dependence matrix;

Compute contribution of final configuration to transition duration;

Compute maximum delays due to departure queue flush-out;

Set transition duration to maximum of contribution of current configuration to transition duration plus maximum delay from exclusive dependence matrix, contribution of final configuration to transition duration, and delays due to departure queue flush-out;

END DUR;

ROUTINE ADJST
[If two configurations in transition have same arrival runways then routine ADJST uses a differenct algorithm to compute transition duration and capacity]

IF departure-departure delay is equal to zero **OR** current departure queue is empty

THEN transition duration is equal to zero and transition hour capacity is equal to zero;

ELSE
Set transition duration equal to maximum of departure-departure delay and departure queue flush-out time;

Set transition hour capacity to capacity of current configuration prorated for duration of transition;

END ADJST;

```
ROUTINE EDP
    [This routine prepares expanded dependence matrix or constraint matrix]

    Compute arrival/arrival quadrant of constraint matrix;

    Compute departure/arrival quadrant of constraint matrix;

    Compute arrival/departure quadrant of constraint matrix;

    Compute departure/departure quadrant of constraint matrix;

END EDP;


FUNCTION OBJFUN
    [This routine computes value of objective function which is transition capacity]

    Set transition capacity to sum of each LP variable multiplied by cost coefficient (fix demand loads);

END OBJFUN;
```

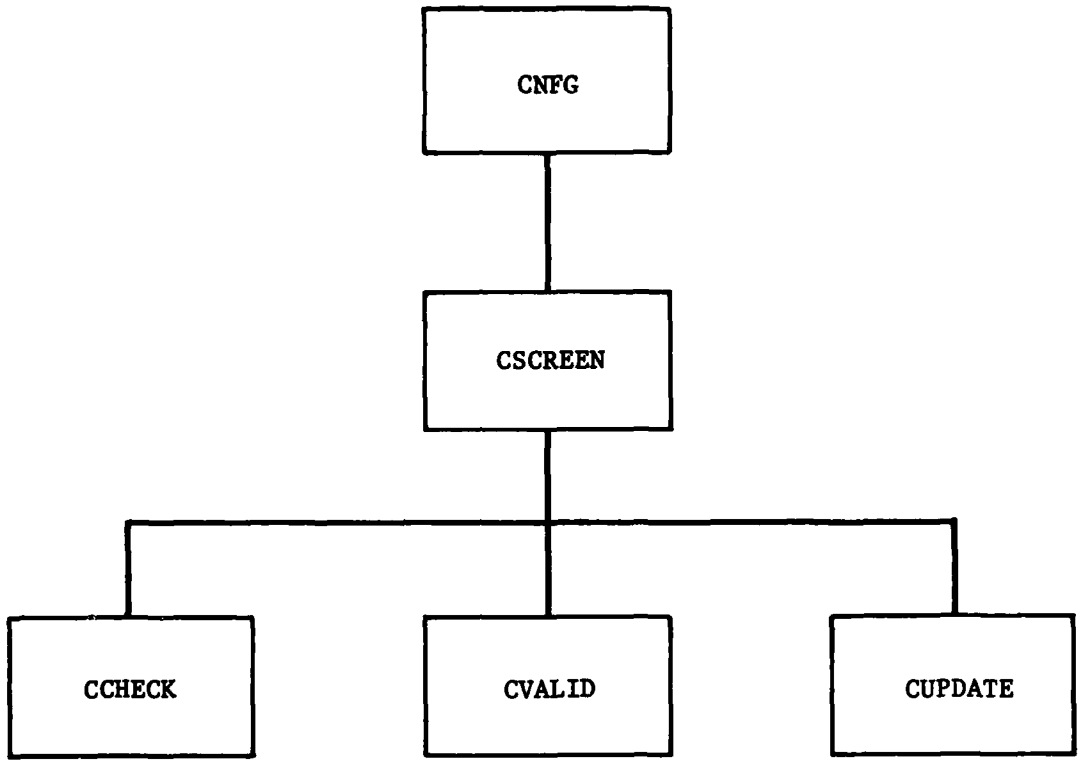


FIGURE A-14
SCHEMATIC DIAGRAM OF
CONFIGURATION INFORMATION SCREEN ROUTINE

```
ROUTINE  CNFG
    [This routine invokes configuration information screen for both current and forecasts environments]

   REPEAT UNTIL  (current program status is not equal to PF12);

       Save a copy of CONFIG structure and CONFIND variable for 'restore previous screen' function;

       REPEAT UNTIL  (current program status is not equal to PF9);

           Switch screens;  [current to forecast and vise versa]

           CALL  CSCREEN;
               [This routine controls configuration information screen]

       ENDREPEAT;

   ENDREPEAT;

   IF  screen message for both environments are equal to 'DATA ENTERED'

       THEN  update CONFIG and CONFIND with new screen entries;

END  CNFG;
```

```
ROUTINE  CSCREEN
    [This routine controls configuration information screen]

    PERFORM  INITIALIZATION;

    PERFORM  SET_UP_SCREEN_POINTERS_(CNFG);

    REPEAT UNTIL  (current program status is not equal to ENTER);

        IF  current operating configuration is eligible

            THEN

                PERFORM  OUTPUT_SET_UP_(TOTAL);

                PERFORM  OUTPUT_SET_UP_(NORTH):

                PERFORM  OUTPUT_SET_UP_(SOUTH);

                PERFORM  OUTPUT_SET_UP_(BALANCING_ARRIVALS);

                PERFORM  OUTPUT_SET_UP_(BALANCING_DEPARTURES);

                PERFORM  OUTPUT_SET_UP_(OTHERS);

            ELSE
                Darken screen;

                Issue message 'THIS CONFIGURATION IS INELIGIBLE;

        PERFORM  DISPLAY_PANEL;

        IF  current program status is equal to PA1

            THEN  stop;

        IF  current program status is equal to ENTER

            THEN
                Set text masks and data masks to normal intensity;
```

```
CALL  CCHECK;
      [This routine checks for errors occurred on screen as a result of an erroneous entry
      and returns value for cursor pointing to first data field where an error has
      occurred; and an appropriate screen message is issued advising user with corrections]

IF  screen message is not equal to 'DATA ENTERED'

     THEN  highlight erroneous entry;

     ELSE

          CALL  CVALID;
                [This routine performs data validation checks on screen entries and
                returns value for cursor pointing to first invalid data field.  Also, an
                appropriate screen message is issued advising user with corrections]

          IF  screen message is not equal to 'DATA ENTERED'

               THEN  darken screen;

               ELSE

                    CALL  CUPDATE;
                          [This routine left-justifies current operating configuration
                          entries]

                    Add current time to screen message;

   ENDREPEAT;

END  CSCREEN;
```

```
PROCESS  INITIALIZATION
    [This process performs a number of necessary initializations for CSCREEN routine]

    Set text and data masks to normal intensity;

    Set screen message data masks to high intensity;

    Set current operating configuration in form of X's on top of configuration information screen;

    Set a number of configuration indicators in form of bit strings for later use in determining warning
    messages on configuration information screen;

END  INITIALIZATION;


PROCESS  SET_UP_SCREEN_POINTERS_(CNFG)
    [This process sets up screen pointers for DMS use]

    Structure CONFIG_LOADLIST is set up with address of location of CNFG_DATA variables;
    [DMS uses CONFIG_LOADLIST to load and unload data onto and from screen]

END  SET_UP_SCREEN_POINTERS_(CNFG);


PROCESS  OUTPUT_SET_UP_(TOTAL)
    [This process sets up screen variable with total airport information]

    IF  both complexes are saturated

        THEN darken screen in area designated to demand balancing information AND issue message 'SATURATED';

        ELSE
            Set total airport percentage of arrivals;
            Set total airport saturation level;
            Set total airport constrained arrival and departure capacities;
            Set total airport arrival and departure demands;

END  OUTPUT_SET_UP_(TOTAL);
```

```
PROCESS  OUTPUT_SET_UP_(NORTH)
    [This process set up screen variable with north complex information]

    Set north complex percentage of arrivals;

    Set north complex saturation level;

    Set north complex arrival and departure capacities;

    Set north complex arrival and departure demands;

END  OUTPUT_SET_UP_(NORTH)


PROCESS  OUTPUT_SET_UP_(SOUTH)
    [This process sets up screen variable with south complex information]

    Set south complex percentage of arrivals;

    Set south complex saturation level;

    Set south complex arrival and departure capacities;

    Set south complex arrival and departure demands;

END  OUTPUT_SET_UP_(SOUTH);
```

```
PROCESS  OUTPUT_SET_UP_(BALANCING_ARRIVALS)
    [This process sets up screen variables with arrival demand balancing information]

    IF  north complex balanced percentage of arrivals is greater than or equal to zero

        THENIF  number of arrival aircraft moved is greater than zero

            THEN
                Set number of arrival aircraft moved;
                Set complex to south;

            ELSEIF  number of arrival aircraft moved is less than zero

                THEN
                    Set number of arrival aircraft moved;
                    Set complex to north;

                ELSE
                    Set 'NO' for number of aircraft moved; darken screen for complex line;

        ELSE  darken screen for arrival balancing information lines;

END  OUTPUT_SET_UP_(BALANCING_ARRIVALS);
```

```
PROCESS  OUTPUT_SET_UP_(BALANCING_DEPARTURES)
    [This process sets up screen variables with departure demand balancing information]

    IF  north complex balanced percentage of arrivals is greater than or equal to zero

        THENIF  number of departure aircraft moved is greater than zero

            THEN
                Set number of departure aircraft moved;
                Set complex to south;

            ELSEIF  number of departure aircraft moved is less than zero

                THEN
                    Set number of departure aircraft moved;
                    Set complex to north;

                ELSE
                    Set 'NO' for number of aircraft moved; darken screen for complex line;

        ELSE  darken screen for departure balancing information lines;

END  OUTPUT_SET_UP_(BALANCING_DEPARTURES);
```

__PROCESS__ OUTPUT_SET_UP_(OTHERS)

[This process sets up screen variable with rest of information needed on configuration information screen]

__IF__ MIDWAY flag is on

__THEN__ set message 'COORDINATE WITH MIDWAY TRAFFIC';

__IF__ HIRL is inoperable on any of current operating configuration's runways

__THEN__ set message '(RUNWAY ID) INELIGIBLE BETWEEN SUNSET & SUNRISE';

__END__ OUTPUT_SET_UP_(OTHERS);

```
ROUTINE  CCHECK
   [This routine checks for errors occurred on screen as a result of an erroneous entry and returns value
   for cursor pointing to first data field where an error has occurred; and an appropriate screen message
   is issued advising user with corrections]

   IF   any of data fields designated to runways in current operating configuration contains entries other
        than 'X ' or blanks

        THEN  issue message 'INPUT MUST BE X OR BLANK';

END  CCHECK;

ROUTINE  CVALID
   [This routine performs data validation checks on screen entries and returns value for cursor pointing to
   first invalid data field.  Also, an appropriate screen message is issued advising user with corrections]

   IF  configuration ID contained on screen is not in list of possible O'Hare configurations

        THEN  issue message 'THIS CONFIGURATION IS NOT KNOWN';

END  CVALID;

ROUTINE  CUPDATE
   [This routine left-justifies current operating configuration entries]

END  CUPDATE;
```

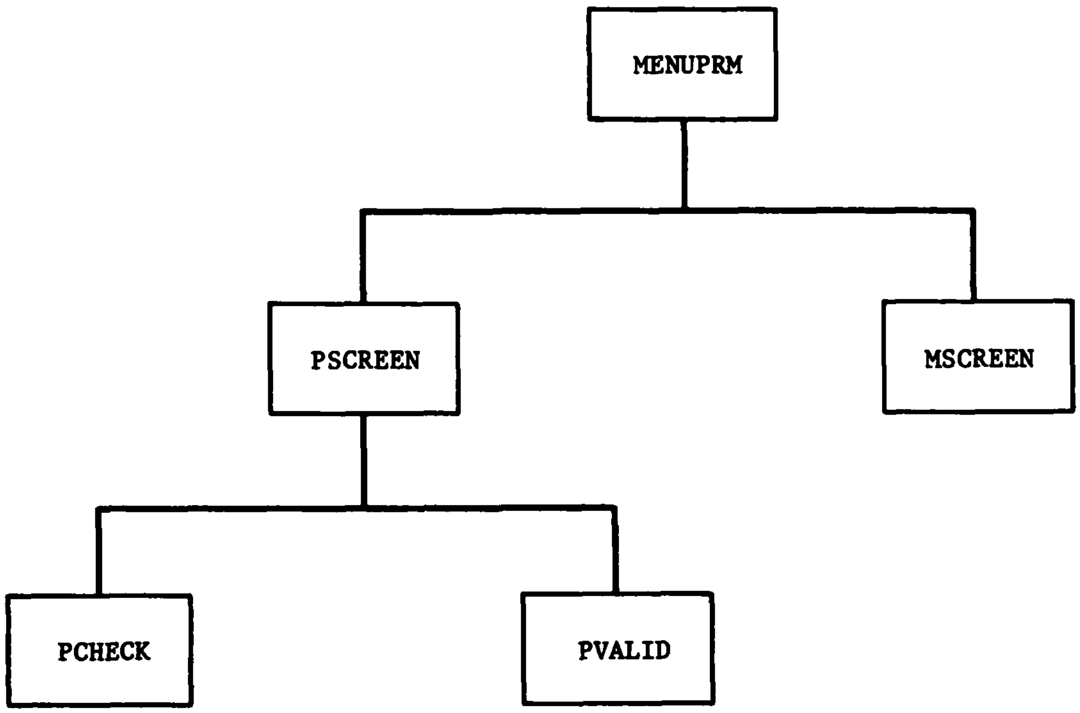


FIGURE A-15
SCHEMATIC DIAGRAM OF
MENU AND PARAMETER SCREENS ROUTINE

```
ROUTINE  MENUPRM
    {This routine invokes menu and/or parameter screen}

    Set screen indicator to menu screen;

    REPEAT UNTIL  (current program status is not equal to PF12);

        IF  screen indicator is set to parameter screen

            THEN  save a copy of PARAM structure for 'restore previous screen' function;

        Switch screens  [menu to parameters or vise versa]

        REPEAT UNTIL  (current program status is not equal to PF11);

            IF  screen indicator is set to menu screen

                THEN CALL  MSCREEN;
                     {This routine controls menu screen}

                ELSE CALL  PSCREEN;
                     {This routine controls parameter screen}

        ENDREPEAT;

    ENDREPEAT;

    IF parameter screen message is equal to 'DATA ENTERED'

        THEN  update PARAM and CNVTPRM with new screen entries;

END  MENUPRM;
```

```
ROUTINE  MSCREEN
    {This routine controls menu screen}

    Set screen pointer for TERMINATION data field;

    REPEAT UNTIL  (current program status is not equal to ENTER) OR (termination indicator is equal to 'X ');

         PERFORM  DISPLAY_PANEL;

         IF  current program status is equal to PA1

              THEN  stop;

    ENDREPEAT;

END  MSCREEN;
```

```
ROUTINE  PSCREEN
    [This routine controls parameter screen]

    Set data masks to normal intensity;

    Set message data mask to high intensity;

    Set cursor to position 1;

    PERFORM  SET_UP_SCREEN_POINTERS_(PARM);

    REPEAT UNTIL  (current program status is not equal to ENTER);

         PERFORM  DISPLAY_PANEL;

         IF  current program status is equal to PA1

              THEN  stop;

         IF  current program status is equal to ENTER

              THEN

                   Set data masks to normal intensity;

                   Set message data mask to high intensity;

                   CALL  PCHECK;
                        [This routine checks for errors occurred on screen as a result of erroneous entry
                        and returns value for cursor pointing to first data field where an error has
                        occurred; and an appropriate screen message is issued advising user with corrections]

                   IF  screen message is not equal to 'DATA ENTERED'

                        THEN  highlight erroneous entry;

                        ELSE

                             CALL  PVALID;
                                  [This routine performs data validation checks on screen entries and
                                  returns value for cursor pointing to first invalid data field.  Also, an
                                  appropriate screen message is issued advising user with corrections]
```

```
            IF  screen message is not equal to 'DATA ENTERED'
                THEN  highlight invalid entry;
                ELSE  add current time to screen message;
    ENDREPEAT;
END  PSCREEN:
```

```
PROCESS  SET_UP_SCREEN_POINTERS_(PARM)
    [This process sets up screen pointers for DMS use]

    Structure PARM_LOADLIST is set up with address of location of PARM_DATA variables;
    [DMS used PARM_LOADLIST to load and unload data onto and from screen]

END  SET_UP_SCREEN_POINTERS_(PARM);
```

```
ROUTINE  PCHECK
   [This routine checks for errors occurred on screen as a result of an erroneous entry and returns value
    for cursor pointing to first data field where an error has occurred; and an appropriate screen message
    is issued advising user with corrections;

    IF  a character data is detected in any of numeric data fields

         THEN  issue message 'NUMERIC INPUT REQUIRED';

    IF  a negative number is detected in any of data fields

         THEN  issue message 'NON_NEGATIVE INPUT REQUIRED';

END  PCHECK;


ROUTINE  PVALID
   [This routine performs data validation checks on screen entries and returns value for cursor pointing to
    first invalid data field.  Also, an appropriate screen message is issued advising user with corrections]

    IF  any of crosswind or tailwind components of wind thresholds exceed 50

         THEN issue messages 'CROSSWIND THRESHOLD MUST NOT EXCEED 50 KNOTS' or 'TAILWIND THRESHOLD MUST NOT
              EXCEED 50 KNOTS';

END  PVALID;
```

APPENDIX B

SYNTAX OF E PSEUDOCODE*

This appendix provides a concise overview of the syntax of the pseudolanguage E. The information supplied should be sufficient to allow the reader to decipher the logic specified in this document. For a complete discussion of pseudolanguage in general and E in particular, see Reference 5.

B.1 GENERAL INFORMATION

A. E = Eclectic System Specification Language

B. E is similar to other pseudolanguages, except that indentation counts: no BEGIN/END, IF/ENDIF, DO/OD, etc.

C. E character set conventions:

Underscored text denotes E constructs.
Uppercase text denotes "real" program statements.
Lowercase text denotes abstract (pseudo) statements.
Brackets ("[", "]") denote comments.
Semicolons are used as statement delimiters.

An example:

```
REPEAT UNTIL (all conditions satisfied);
          Obtain message type;
          IF (obsolete message OR A EQ SQRT(B))
                    THEN PERFORM message_elimination;
ENDREPEAT;
```

D. Identifiers have no inherent length limit. Underscores are used to break up long names, as shown in the example above.

* Source: Reference 4.

B.2 BLOCKS

External Blocks

TASKs and ROUTINEs are the external blocks supported by E. Although they are functionally equivalent, ROUTINEs tend to be subordinate to (i.e., invoked by) TASKs.

Syntax:

```
TASK taskname
     [IN (input parameter(s))]
     [OUT (output parameter(s))]
     [INOUT (modified parameter(s))];
...
...
END taskname;
```

```
ROUTINE routinename
     [IN (input parameter(s))]
     [OUT (output parameter(s))]
     [INOUT (modified parameter(s))];
...
...
END routinename;
```

Input parameters are read by not modified; output parameters are set by the block; modified parameters are read and then modified.

Functions may also be defined. The returned value may be assigned to the single output parameter or, alternatively, assigned to the function name itself.

```
FUNCTION functionname
     [IN (input parameter(s))]
     [OUT (output parameter)];
...
...
END functionname;
```

Invocation of External Blocks

TASKs and ROUTINEs:

```
CALL blockname
     [IN (input parameter(s))]
     [OUT (output parameter(s))]
     [INOUT (modified parameter(s))];
```

Functions are invoked by name:

```
J = SQRT(K);
L = OWNER_OF(M);
```

Internal Blocks

Internal blocks (known as processes) serve as a means of decomposing a large block (external or internal) into manageable segments. They are known only to the block in which they are defined. They do not accept parameters, as it is assumed that internal blocks have access to all variables known to the invoking block.

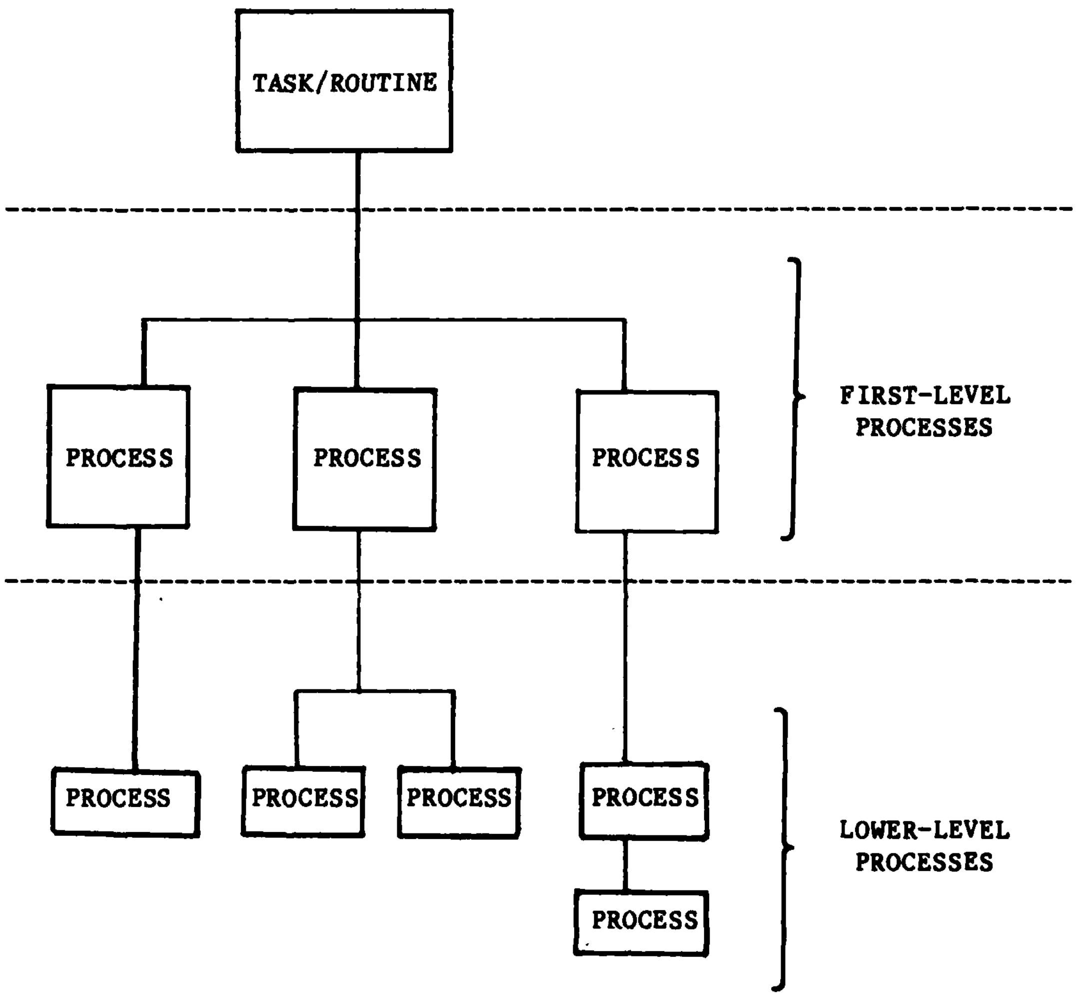
```
PROCESS processname;
 ...
 ...
END processname;
```

Invocation of Internal Blocks

PERFORM processname;

Nomenclature of Internal Blocks

A TASK or ROUTINE might be decomposed into processes as follows:



Processes frequently invoke external blocks (TASKs and ROUTINESs).

B.3 DATA TYPES

Variables

Variables are declared at the beginning of a block in the form shown below:

```
BIT bitname;                        single logical bit
BITS stringname;                    bit string
CHR stringname;                     character string
FLG bitname;                        synonym for BIT
FLT name;                           floating point number
INT name;                           integer
PTR name;                           pointer
```

The precision of numeric variables and the length of strings are implementation-dependent, although comments on the declaration may be used to indicate specific requirements.

Arrays are declared by means of subscripts:

```
INT ZONES(4);                       four-element array
```

Constants

Constants in E are variables that are assigned a value when they are declared and keep that value forever. By convention, constant names are preceded by dollar signs in E to remind the reader that they are special.

```
FLT $FTPERNM = 6076.115;
INT $TL      =    2;                Turn Left
```

On occasion, the constant is shown without a corresponding value. This convention indicates that a constant is required but that its value may be anything the system implementor chooses.

Built-in Constants

Strictly speaking, the only hard constants permitted in code are zero and one. E recognizes the logical constants $TRUE and $FALSE. Two special statements are provided to set and clear bits:

```
    SET bitname;                        bitname = $TRUE
    CLEAR bitname;                      bitname = $FALSE
```

The built-in constant $NULL defines a null pointer.

Data Structures

E provides a mechanism for grouping related variables into data structures:

```
    STRUCTURE structurename
         GROUP groupname
              FLT variablename
                   ...
                   ...
    ENDSTRUCTURE;
```

An arbitrary number of groups may be defined. The keyword LIKE may be used to indicate that a group is identical to another group or a structure identical to another structure.

When a variable that has been declared inside a data structure is used in code, it must be qualified with the name of the structure (and the name of the group, if needed to resolve ambiguity):

```
    SVECT.X = PREC.ctl_thresh.ALT;
```

When groups are manipulated as a unit, the GROUP keyword may be included as an aid to the reader:

```
    CALL COMPUTE
         INOUT (GROUP SVECT.radar_reports);
```

Expressions

E assumes the existence of the usual repertoire of built-in functions (ABS, SIN, SIGN, ...). Within logical expressions, logical operators are of the form LE, LT, GE, and so on.

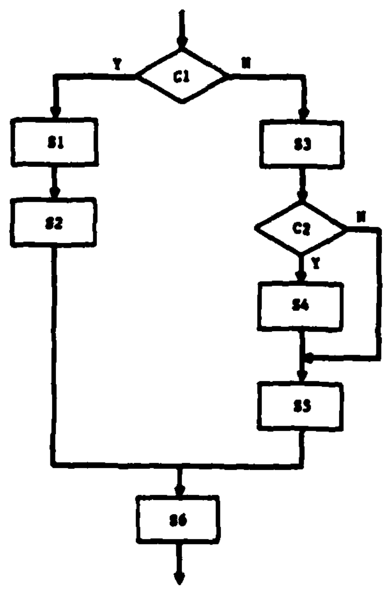
B.4 FLOW-OF-CONTROL CONSTRUCTS

E uses a set of flow-of-control constructs that incorporates structured programming principles. Readers familiar with othe pseudolanguages are once again reminded that indentation counts.

IF-THEN-ELSE

This is the usual conditional. The ELSE clause is optional (but recommended in complex statements). Since readers will presumably be familiar with the syntax of this construct, the example below is meant to emphasize the effects of indentation.

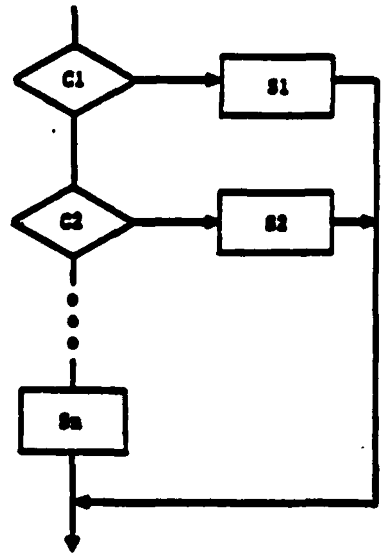
```
    IF (cond1)

         THEN s1;

              S2;

         ELSE s3;

              IF (cond2)

                   THEN s4;

              S5;

    S6;
```



IF-ELSEIF-OTHERWISE

This is the multiple-choice conditional (like SELECT-CASE in other languages). The conditions are mutually exclusive. If all the logical tests fail, the optional OTHERWISE clause is executed.

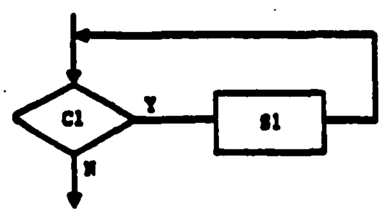
```
IF (cond1)
     THEN s1;
ELSEIF (cond2)
     THEN s2;
     ...
     ...
[OTHERWISE sn;]
```



REPEAT-WHILE

This is the first of threee looping contructs. Note that the logical test takes place at the top of the loop, so that the loop may never be executed.

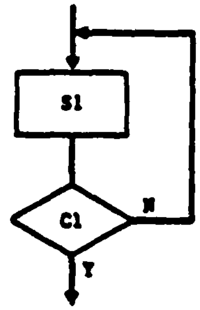
```
REPEAT-WHILE (cond1);
     S1;
     ...
ENDREPEAT;
```



REPEAT-UNTIL

This construct is the complement of REPEAT-WHILE: the logical test is performed at the end of the loop and the loop continues while the condition is not true. The loop body is always executed at least once.

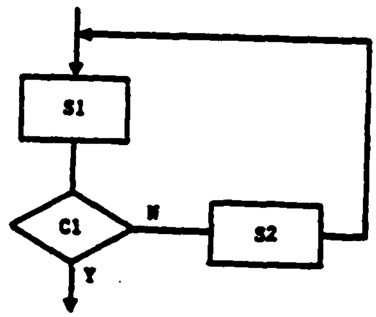
```
REPEAT UNTIL (cond1);
      S1;
      ...
ENDREPEAT;
```



LOOP-EXITIF-ENDLOOP

This construct provides a good general-purpose looping mechanism.

```
LOOP;
      S1;
EXITIF (cond1);
      S2;
ENDLOOP;
```



In some cases, low-level operations within the three looping constructs (such as obtaining the next element in a linked list) will be omitted for brevity.

APPENDIX C

UTILITY PROGRAMS IN CMS

The following Appendix presents a number of utility programs that are used repeatedly in the CMS software. These programs are presented in the form of pseudocodes:

```
FUNCTION M
    [This function accepts as an input a data item containing a particular category of runway visibility
    minimum and converts it from RVR to miles]

END M;
```

```
FUNCTION  M
    [This function accepts as an input a variable containing RVR reading and converts it to miles]
    IF  X GT 2.0
        THENIF  X LT 18.0
            THEN  X = .25;
            ELSEIF  X LT 32.0
                THEN  X = .5;
                ELSEIF  X LT 45.0
                    THEN  X = .63;
                    ELSEIF  X LT 45.0
                        THEN  X = .75;
                        ELSEIF  X LT 50.0
                            THEN  X = .88;
                            ELSEIF  X LT 60.0
                                THEN  X = 1.0;
                                ELSEIF  X LT 78.0
                                    THEN  X = 1.25;
                                    ELSE  X = 2.0;
END  M;
```

```
FUNCTION X
    [this function returns a flag value; if entry on screen is blank, ' x', or 'x ', then flag = 0 is
    returned, any other entry sets flag = 1, also, entry 'x' is left-justified on return]

END X;
```

```
FUNCTION  X

    OUT  (FLAG);

    INOUT  (X);
        [This function returns a flag value; if entry on screen is blank, ' X', or 'X ' then flag = 0 is
        returned, any other entry sets flag = 1.  Also, entry 'X' is left justified on return]

    CHR  X  [character variable of length 2 representing entry on screen]

    INT  FLAG  [integer flag indicating whether screen entry is correct]

    FLAG = 1;

    IF  (X EQ ' X' OR X EQ 'X ')

        THEN

            FLAG = 0;

            X = ' X';  [left justified]

            ELSEIF  (X EQ '  ')

                THEN  FLAG = 0;

END X;
```

```
FUNCTION  F
    [This function accepts a numerical floating point data, and after rounding it off to nearest integer,
    converts it to character data]

END  F;
```

```
FUNCTION F

    IN (N,X);

    OUT (CSTRING);
        [This function accepts a numerical floating point data, and after rounding it off to nearest
        integer, converts it to character data]

    CHR C(N) [character variable of length 1 to store each digit as character data]

    CHR DIGITS (0:9)   [character variable of length 1 stores all numerical digits in character form, i.e.,
                        DIGITS(0) = ' ', DIGITS(1) - '1', etc.]

    BIT FLAG [initialized to '0'B]

    Y = X + .5; [round off to nearest integer]

    LOOP; [L = 1 TO N]

        EXP = 10.0 ** (N-L);

        K = FLOOR (Y/EXP);

        IF (K EQ 0) AND (FLAG EQ '0'B) AND (L NE N)

            THEN C(L) = ' ';   [places leading blanks if any exist]

            ELSE [places individual digits]

                FLAG = '1'B;
                C(L) = DIGITS(K);
                Y = Y-K*EXP;

    ENDLOOP;

    CSTRING = STRING(C);

END F;
```

```
FUNCTION  GMT
    [This function returns character representation of current running time converted to Greenwich Mean Time]

END  GMT;
```

```
FUNCTION  GMT

    OUT  (GMT);
         [This function returns character representation of current running time converted to Greenwich Mean
         Time]

    INT  0  [interger representing time difference between local time and Greenwich Mean Time initialized to
            4]

    CHR  TABLE(0:23)  [character representation of day's 24 hours, i.e., TABLE(0) = '00', TABLE(1) =
                    '01', etc.]

    T = SUBSTR (TIME, 1,4);

    HOUR = SUBSTR(T,1,2);  [hour portion of time is extracted]
    MIN  = SUBSTR(T,3,2);  [minute portion of time is extracted]

    FLAG = '0'B;

    REPEAT UNTIL  (FLAG EQ  '1'B); [J = 0 TO 23]

         IF  TABLE(J) = HOUR

              THEN  FLAG = '1'B

    ENDREPEAT;

    IF   (J+0) LT  24  [Actual conversion to GMT]

         THEN  GMT = TABLE(J+0) CONCATENATE MIN;

         ELSE  GMT = TABLE(J+0-24) CONCATENATE MIN:

END  GMT;
```

APPENDIX D

PL/I BUILT-IN FUNCTIONS USED IN CMS

The following PL/I built-in functions are used in the CMS software:

ABS(X) - ABS returns the absolute value of a given expression X.

ADDR(X) - ADDR returns a pointer value that identifies the location at which a given variable X has been allocated.

CEIL(X) - CEIL returns the smallest integer greater than or equal to a given value X.

FLOAT(X) - FLOAT returns the floating point representation of a given value X.

FLOOR(X) - FLOOR returns the largest integer less than or equal to a given value X.

INDEX(X_1, X_2) - INDEX returns a halfword binary integer indicating the starting position within the string X, of a substring identical to string X_2.

MOD (X_1, X_2) - MOD returns the smallest non-negative value, R, such that:

$(X_1 - R)/X_2 = n$ where n is an integer.

SUBSTR(X_1, X_2, X_3) - SUBSTR returns a substring of the given string X_1.

X_1 - string from which the substring is to be extracted.

X_2 - an expression that can be converted to integer indicating the starting position of the substring in X_1.

X_3 - an expression that can be converted to integer specifying the length of the substring in X_1.

TIME - TIME returns a characterstring of length nine, in the form hhmmssttt, where:
hh - the current hour
mm - number of minutes
ss - number of seconds
ttt - number of milliseconds

TRANSLATE(X_1, X_2, X_3) - TRANSLATE returns a string the same length as a given string X_1, where all or some of the characters may have been changed. Characters are changed according to a look-up table provided by strings X_2 and X_3.

VERIFY (X_1, X_2) - VERIFY returns a default precision fixed point binary integer indicating the position in the given string X_1 of the first character or bit that is not in the given string X_2.

APPENDIX E

DATA BASE FORMAT

This appendix presents the format associated with the CMS data base. Figure E-1 contains the CMS data base as follows:

- Line A - Times at which data from each screen was stored. Times are expressed in hours and minutes, Greenwich Mean Time (format: A4).
- Line B - Midway flag for both current and forecast versions of airport status screen (format 2A2); configuration index, current and forecast (format 2F); departure queue lengths for up to four runways in both numeric and character forms (format 4A2 and 4F2.0).
- Line C - Crosswind and tailwind thresholds, in knots, in both numeric and character form (format 4A4 and 4F4.1).
- Line D - Airport status screen replica for both current and forecast conditions in character form (format A).
- Line E - airport status screen information for both current and forecast conditions in numerical form (format 8F6.1).
- Line F - Equipment status screen replica for both current and forecast conditions in character form (format A).
- Line G - Current demand profile screen information in character format (format 13A4).
- Line H - Forecast demand profile screen information in character form (format 13A4).
- Line I - Current and forecast demand profile screen information in numerical form (format 26F5.1).
- Line J - Equipment planning log screen replica in character form (format A).

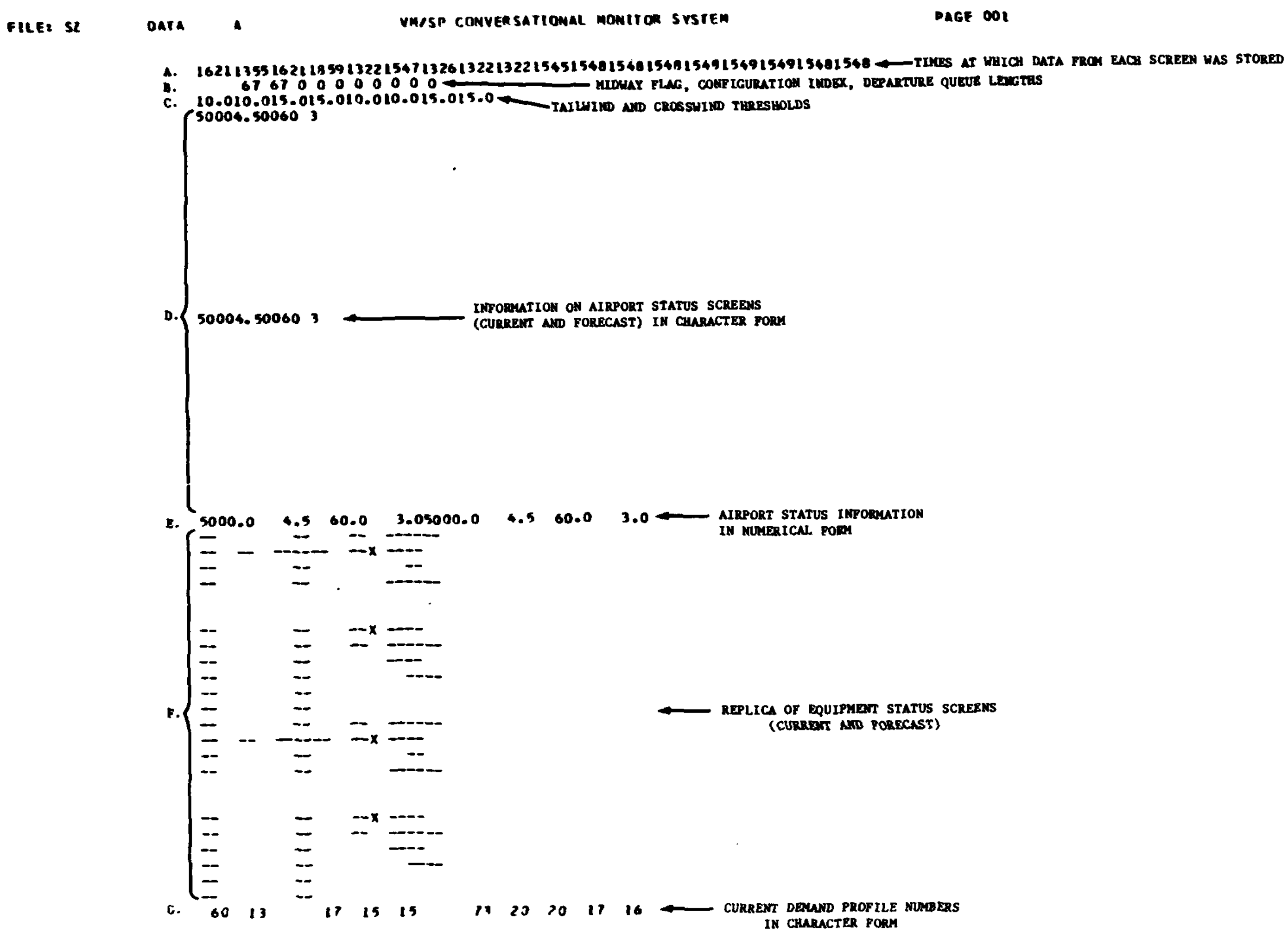


FIGURE E-1
DATA BASE FORMAT

```
FILE: SZ        DATA     A                 VM/SP CONVERSATIONAL MONITOR SYSTEM                     PAGE 002

H.   48  10       15  11  12       49  12  12  14  11    <---- FORECAST DEMAND PROFILE NUMBERS
                                                                IN CHARACTER FORM
I. { 60.0 13.0  0.0 17.0 15.0 15.0  0.0 73.0 20.0 20.0 17.0 16.0  0.0  <---- DEMAND PROFILE NUMBERS IN NUMERICAL FORM
   { 48.0 10.0  0.0 15.0 11.0 12.0  0.0 49.0 12.0 12.0 14.0 11.0  0.0
   { 22RHIRL       12002000
   { 32LALS        22002315REPAIRS
J. {                                              <---- INFORMATION ON EQUIPMENT PLANNING LOG SCREEN
   {                                                          IN CHARACTER FORM
K. { 1200200022002315999999999999999999999999999999999999999999999999
   { 99999999999999999999999999999999999999999999999999999999   <---- IN NUMERICAL F?M
   { 1200 5000 4.50 060     3 SCATTERED CLOUDS
L. {                                              <---- INFORMATION ON WEATHER AND WIND PLANNING LOG SCREEN
   {                                                          IN CHARACTER FORM
   { 1200 5000  4.50   60    39999    0  0.00    0    09999    0  0.00   0    0
   { 9999    0  0.00    0    09999    0  0.00    0    09999    0  0.00   0    0
M. { 9999    0  0.00    0    09999    0  0.00    0    09999    0  0.00   0    0  <---- IN NUMERICAL FORM
   { 9999    0  0.00    0    09999    0  0.00    0    09999    0  0.00   0    0
   { 9999    0  0.00    0    0
N. {                                              <---- INFORMATION ON RUNWAY CONDITION PLANNING LOG SCREEN
   {                                                          IN CHARACTER FORM
O.   9999999999999999999999999999999999999999999999999999
   { 0000 59 40  6 17 20 16 20 15 15 10
P. { 0100 40 40 10 10 10 10 10 10 10 10   <---- DEMAND PLANNING LOG INFORMATION
   { 0200 40 40 10 10 10 10 10 10 10 10         (CHARACTER AND NUMERICAL FORM)
```

FIGURE E-1
DATA BASE FORMAT
(CONTINUED)

DEMAND PLANNING LOG INFORMATION CONTINUED

P.	0300	40	40	10	10	10	10	10	10	10	10
	0400	40	40	10	10	10	10	10	10	10	10
	0500	40	40	10	10	10	10	10	10	10	10
	0600	40	40	10	10	10	10	10	10	10	10
	0700	40	40	10	10	10	10	10	10	10	10
	0800	40	40	10	10	10	10	10	10	10	10
	0900	40	40	10	10	10	10	10	10	10	10
	1000	40	40	10	10	10	10	10	10	10	10
	1100	40	40	10	10	10	10	10	10	10	10
	1200	64	50	18	21	14	11	15	15	10	10
	1300	63	71	18	21	14	10	20	21	15	15
	1400	44	56	5	19	10	10	15	15	16	10
	1500	75	73	21	22	17	15	15	23	15	20
	1600	57	72	11	16	15	15	21	19	17	15
	1700	46	43	10	15	10	11	10	10	13	10
	1800	72	66	20	25	15	12	18	17	15	16
	1900	66	67	21	22	13	10	18	21	18	10
	2000	69	58	23	23	13	10	18	20	10	10
	2100	51	61	15	15	11	10	20	15	16	10
	2200	95	78	20	32	23	20	20	20	20	18
	2300	40	40	10	10	10	10	10	10	10	10
Q.	0	59	60	6	17	20	16	20	15	15	10
	100	40	40	10	10	10	10	10	10	10	10
	200	40	40	10	10	10	10	10	10	10	10
	30C	40	40	10	10	10	10	10	10	10	10
	400	40	40	10	10	10	10	10	10	10	10
	500	40	40	10	10	10	10	10	10	10	10
	600	40	40	10	10	10	10	10	10	10	10
	700	40	40	10	10	10	10	10	10	10	10
	800	40	40	10	10	10	10	10	10	10	10
	900	40	40	10	10	10	10	10	10	10	10
	1000	40	40	10	10	10	10	10	10	10	10
	1100	40	40	10	10	10	10	10	10	10	10
	1200	64	50	18	21	14	11	15	15	10	10
	1300	63	71	18	21	14	10	20	21	15	15
	1400	44	56	5	19	10	10	15	15	16	10
	1500	75	73	21	22	17	15	15	23	15	20
	1600	57	72	11	16	15	15	21	19	17	15
	1700	46	43	10	15	10	11	10	10	13	10
	1800	72	66	20	25	15	12	18	17	15	16
	1900	66	67	21	22	13	10	18	21	19	10
	2000	69	58	23	23	13	10	18	20	10	10
	2100	51	61	15	15	11	10	20	15	16	10
	2200	95	78	20	32	23	20	20	20	20	18
	2300	40	40	10	10	10	10	10	10	10	10

FIGURE E-1
DATA BASE FORMAT
(CONCLUDED)

o Line K - OTS and RTS values of equipment planning log screen in numerical form (format 30F4.0). These values represent times in hours and minutes, GMT.

o Line L - Weather and wind planning log screen replica in character form (format A).

o Line M - Weather and wind planning log screen information in numerical form.

o Line N - Runway surface conditions planning log screen replica in character form (format A).

o Line O - Runway surface conditions planning log information in numerical form (format 13F4).

o Line P - Demand planning log information in character form. The first column represents time (GMT); the next two columns are total arrival demand and total departure demand, respectively, for that hour. Arrival demand data for each of the four principal arrival fixes is shown in the next four columns, followed by departure demand for each of the four departure directions.

o Line P - Demand planning log information in character form continued.

o Line Q - Demand planning log information in numerical form.

In addition to the CMS data base, there are several CMS permanent data files that are read into the memory at the time of the program execution. Each of these permanent data files is read into a data structure designated to that file. In order to obtain the format of these files, it suffices to look at the description of the data structure associated with that file. These data structures are described in the Volume II, section 2.1. The following table serves as the cross-reference between CMS permanent data files and their data structures.

FILE	DATA STRUCTURE	LOCATION
CAPACITY	CAPFILE	Page 2-5
TRAVEL	FIXTRAV	Page 2-5
RNWYMIN	RWYMIN	Page 2-6
CNFGREQ	CNFGRQ	Page 2-3
OAGDMND	OAGDEM	Page 2-4
DEPEND	DEPMAT	Page 2-5

APPENDIX F

REFERENCES

1. R. L. Fain, "Overview of the O'Hare Runway Configuration Management System," MTR-81W235, The MITRE Corporation, September 1981.

2. S. Kavoussi and M. Segal, "User's Reference Guide for the O'Hare Runway configuration Management System," WP-81W536, The MITRE Corporation, September 1981.

3. R. L. Fain, S. Kavoussi, and G. Scot, "Enhancements to the O'Hare Configuration Management System," Volume I, WP-80W997, The MITRE Corporation, December 1980.

4. R. H. Lentz, et. al., "Automatic Traffic Advisory and Resolution Service (ATARS) Algorithms Including Resolution - Advisory - Register - Logic," MTR-81W120, 1 and 2, The MITRE Corporation, June 1981.

5. H. R. Bulterman, "All About E," WP-81W00552, The MITRE Corporation, October 1981.

6. "OS PL/I Checkout and Optimizing Compilers: Language Reference Manual," GC33-0009-4, IBM, October 1976.

7. "MWCC: User's Guide," ML-81W00001, The MITRE Corporation, August 1981.

8. "IBM VM/SP Display Management System for CMS: Guide and References," SC24-5198-1, IBM, December 1981.

END

FILMED

5-83

DTIC